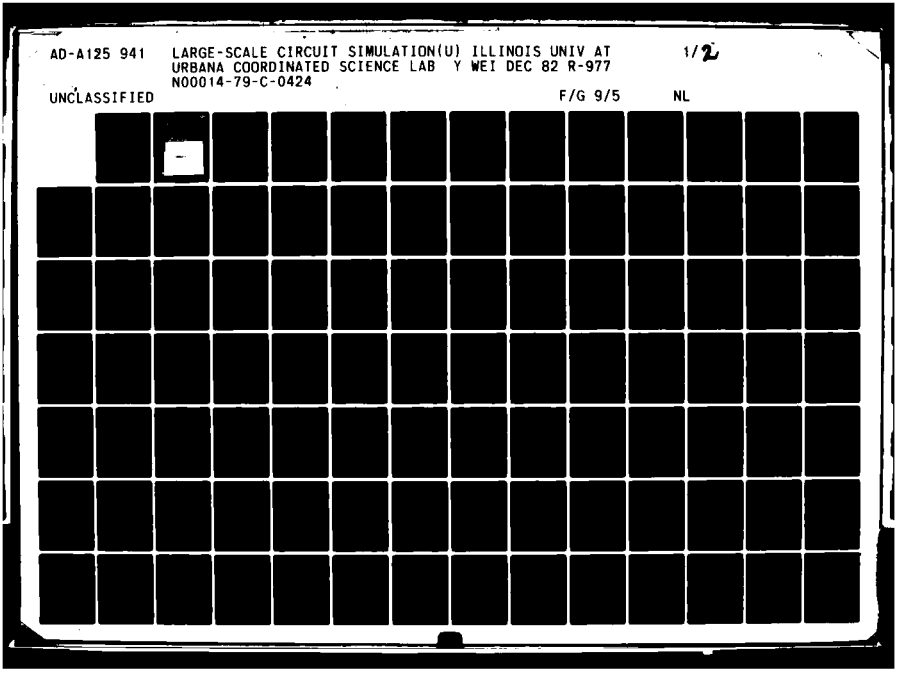
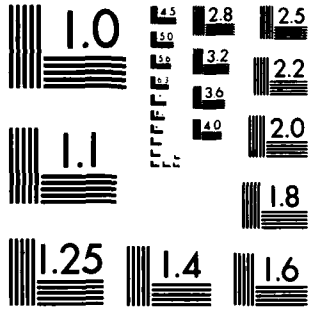
AD-A125 941 LARGE-SCALE CIRCUIT SIMULATION(U) ILLINOIS UNIV AT URBANA COORDINATED SCIENCE LAB Y WEI DEC 82 R-977 N00014-79-C-0424 1/2

UNCLASSIFIED F/G 9/5 NL

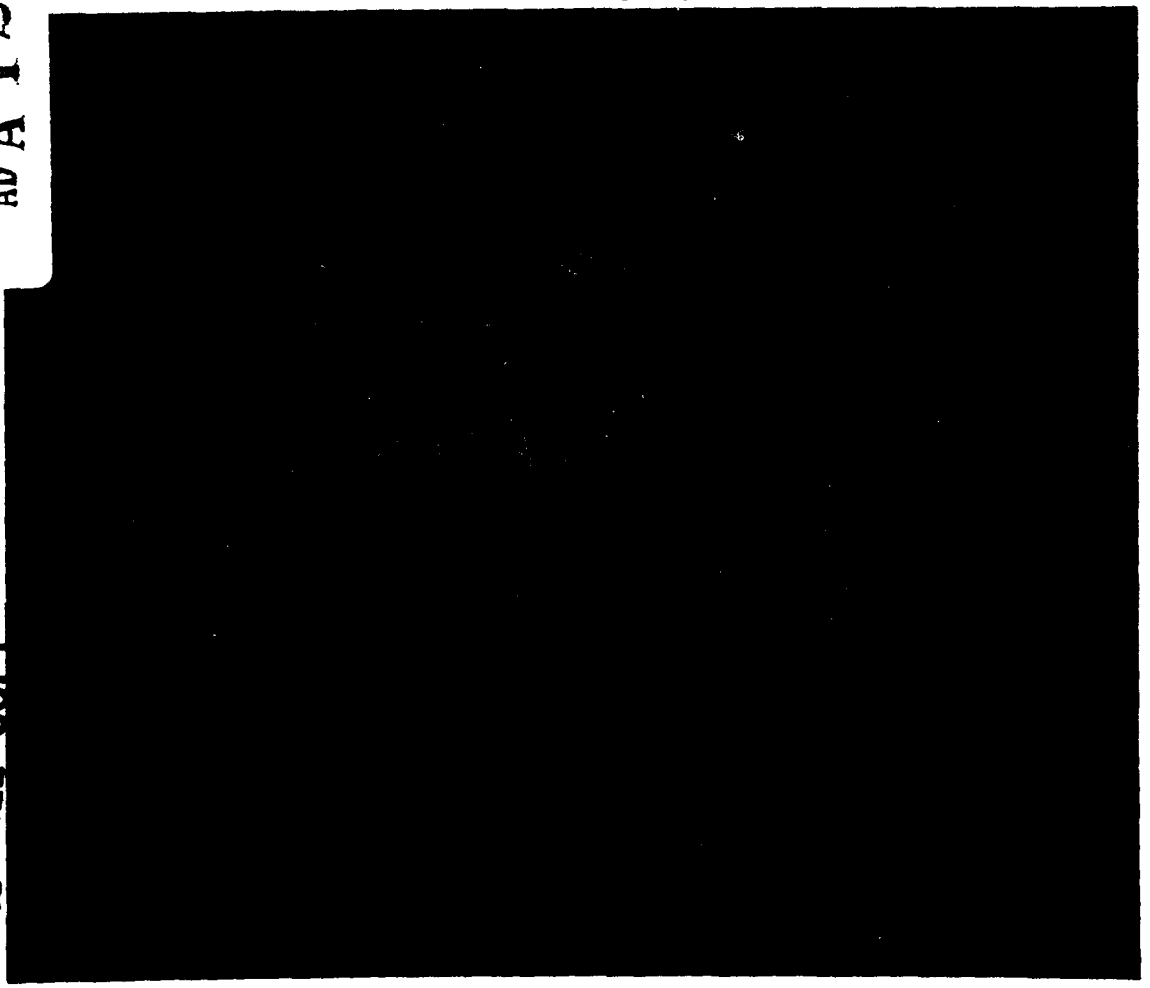




REPORT R-977 DECEMBER 1982 UILU-ENG 82-2243

COORDINATED SCIENCE LABORATORY

LARGE-SCALE CIRCUIT SIMULATION



UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. A125941 / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) LARGE-SCALE CIRCUIT SIMULATION	5. TYPE OF REPORT & PERIOD COVERED Technical Report
	6. PERFORMING ORG. REPORT NUMBER R-977 UILU-ENG 82-2243
7. AUTHOR(s) You-Pang Wei	8. CONTRACT OR GRANT NUMBER(s) N00014-79-C-0424
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory University of Illinois at Urbana-Champaign Urbana, Illinois 61801	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Joint Services Electronics Program	12. REPORT DATE December 1982 / 13. NUMBER OF PAGES 166
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Integrated circuit
Time-domain Simulation of MOS LSI circuits
Analysis sequencing
Modified Gauss - Seidel Method

(very Large Scale Integration) (Large Scale Integration)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The simulation of VLSI circuits falls beyond the capabilities of conventional circuit simulators like SPICE. On the other hand, conventional logic simulators can only give the results of logic levels "1" and "0" with the attendent loss of detail in the waveforms. The aim of developing large-scale circuit simulation is to bridge the gap between conventional circuit simulation and logic simulation.

This research is to investigate new approaches for fast and relatively accurate time-domain simulation of MOS, LSI, and VLSI circuits. New techniques and new algorithms are studied in the following areas: (1) analysis sequencing (2) nonlin-

DD FORM 1 JAN 73 1473

(metal oxide Semiconductor)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20.

ear iteration (3) modified Gauss-Seidel method (4) latency criteria and timestep control scheme. The developed methods have been implemented into a simulation program PREMOS which could be used as a design verification tool for MOS circuits.

LARGE-SCALE CIRCUIT SIMULATION

BY

You-Pang Wei

This work was supported by the Joint Services Electronics Program under Contract N00014-79-C-0424.

LARGE-SCALE CIRCUIT SIMULATION

BY

YOU-PANG WEI

B.S., National Taiwan University, 1975
M.S., University of Illinois, 1979

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1983

Urbana, Illinois

LARGE-SCALE CIRCUIT SIMULATION

You-Pang Wei, Ph.D.
Department of Electrical Engineering
University of Illinois at Urbana-Champaign, 1983

The simulation of VLSI circuits falls beyond the capabilities of conventional circuit simulators like SPICE. On the other hand, conventional logic simulators can only give the results of logic levels "1" and "0" with the attendant loss of detail in the waveforms. The aim of developing large-scale circuit simulation is to bridge the gap between conventional circuit simulation and logic simulation.

This research is to investigate new approaches for fast and relatively accurate time-domain simulation of MOS LSI and VLSI circuits. New techniques and new algorithms are studied in the following areas: (1) analysis sequencing (2) nonlinear iteration (3) modified Gauss-Seidel method (4) latency criteria and timestep control scheme. The developed methods have been implemented into a simulation program PREMOS which could be used as a design verification tool for MOS circuits.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor I. N. Hajj, for his valuable guidance and continual encouragement during the course of this research. He provided good advice in both life and academic work throughout this period. Also, I would like to thank Professor T. N. Trick, my co-advisor, for the many helpful discussions and suggestions in this research. I also feel grateful to Professor W. K. Perkins for his being a member of my dissertation committee and his support. Finally, I wish to thank my wife Tsu-Jane for her support in all respects during this period. Without her encouragement and understanding, it would have been impossible for me to finish my Ph.D. study.

TABLE OF CONTENTS

CHAPTER 1

Introduction

Improvements in semiconductor processing have actually accelerated the complexity of VLSI chips which potentially have hundreds of thousands of transistors. To deal with this complexity two concepts are generally applied: decomposition, which is the process of breaking a problem into manageable pieces, and abstraction, which is the technique of hiding unnecessary detail. Applying these two principles in VLSI design results in a multi-level, hierarchical approach to the design of a complex chip [1]. The set of design verification tools corresponding to the levels of the hierarchy is shown in Fig. 1.1 [2].

Functional simulators are used at the initial design phase to verify the algorithms of the digital system to be implemented. Once the design meets these criteria for the behavioral completeness, an RTL (Register Transfer Level) simulator could be used to verify the potential implementation of the structure. With each RTL module further partitioned into low-level logic building blocks, the logic design is validated by a logic simulator such as MOSSIM or SALOGS [3,4].

The gate level design is implemented into integrated circuits by transistors and associated interconnections. For the analysis of small circuit blocks, circuit simulators, such as SPICE2 [5], have

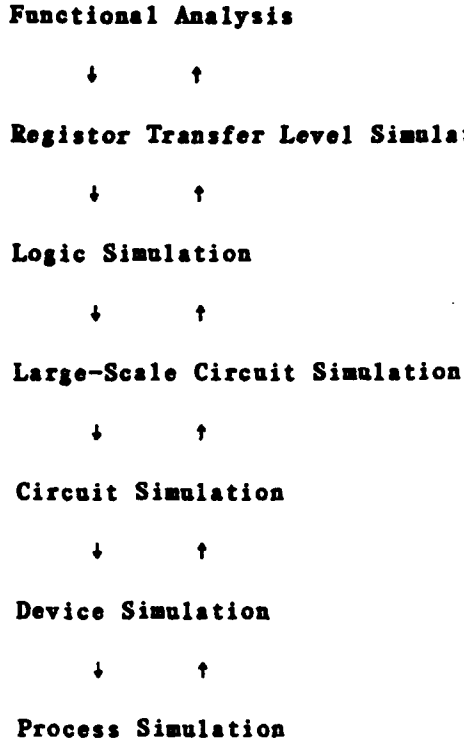


Fig. 1.1 The Hierarchy of Design Verification Tools.

proved effective by providing accurate voltage and current waveforms. The larger blocks may be analyzed in less detail by using a 'large-scale' circuit simulator such as MOTIS, MOTIS-C, or MOSTAP [6,7,8].

For VLSI design, there are some major constraints such as die size, speed and power, which are taken into consideration at each level, often forcing a designer to backtrack when a constraint cannot be met at a lower level. A number of simulations are required before the design is completed. The cost of simulation is expensive, especially for circuit simulation which can accurately predict circuit performance. As the size of the circuit increases, eventually the

cost of conventional circuit analysis becomes prohibitive. Therefore, the large-scale circuit analysis in which some relaxation techniques are used, is developed to reduce the execution time and memory requirements but still provide adequate information about circuit performance.

This research is concerned with the development of numerical methods and scheduling techniques for fast and relatively accurate time-domain simulation of MOS LSI and VLSI circuits. The goal is that the developed methods and techniques could be implemented in a simulator which could be used as a design verification tool for MOS circuits.

The basic approach in most 'large-scale' circuit simulators is, firstly, the partitioning of the circuit into smaller subcircuits, and then, the analysis of these subcircuits in a certain sequence [8, 9, 10, 11, 12]. By using analysis sequencing or selective trace techniques, one may take advantage of the latency properties of the subcircuits in both time and space to reduce the computation time [13, 14]. In this research, MOS circuits are decomposed into 'one-way' subcircuits in the DC sense [3, 8] (i.e., the circuit is assumed to be in steady-state with its capacitors open-circuited). In practice, this partitioning approach produces subcircuits of relatively small sizes and sparse matrix techniques are not necessary. By properly ordering the circuit variables, the circuit equations can be put in an 'almost' lower block triangular form with the upper triangular nonzero terms accounting for any feedback that might exist among the

subcircuits. Traditionally, the Gauss-Seidel method has been used to decouple the feedback terms by assigning previous values to the current 'unsolved' variables [8, 15]. However, this approach suffers from accuracy problems. Furthermore, as will be shown later in this thesis, when floating capacitors exist among the subcircuits, the Gauss-Seidel method will not be consistent and thus not convergent. A 'modified' Gauss-Seidel-Newton method is then introduced to solve the circuit equations and to decouple the feedback terms during the analysis process. The proposed technique, which is based on the use of a forward predictor to estimate the values of the yet unsolved variables in feedback loops, is more accurate than the standard Gauss-Seidel method, without requiring much additional computation. This modified Gauss-Seidel method is shown to be consistent, stable and convergent. As far as analysis sequencing is concerned, a procedure is described in this dissertation, which is different from ones previously proposed in that it schedules only those 'relevant' subcircuits that directly or indirectly affect the output. This approach is combined with a latency technique to further increase the speed of simulation.

An experimental program PREMOS (PREdiction-based simulator for MOS circuits) is developed to implement and test the new algorithms and new schemes in this research. This program is mainly for the time-domain analysis of VLSI MOS digital circuits. It has been shown that PREMOS can produce simulated results whose accuracy is close to that of conventional circuit simulation, whereas the computational

speed is generally within the range of five times slower than that of timing simulators such as MOTIS-C [6, 7]. The speed and circuit-size capability of MOTIS-C have been claimed to be over two orders of magnitude greater than SPICE2 [7]. Hence, PREMOS also has much greater speed and circuit-size capability than SPICE2.

Chapter 2 describes briefly the analysis techniques used in conventional circuit analysis and large-scale circuit analysis. In Chapter 3 the analysis sequencing procedures are explained and the idea of scheduling only 'relevant' parts is studied. The nonlinear DC analysis methods adopted in the solution algorithms are discussed in Chapter 4. Chapter 5 introduces the modified Gauss-Seidel method and provides the numerical study of the method. In Chapter 6, latency criteria and a timestep control scheme are described. Chapter 7 describes the structure of the program PREMOS and gives some simulation examples. In the final chapter, Chapter 8, the conclusions are presented and the areas for future work are pointed out.

There are five appendices. Appendix 1 describes MOS device modeling and capacitor modeling. Appendix 2 contains the input descriptions for circuit elements and their models in the program PREMOS. The control commands and the data structure used in PREMOS are given in Appendix 3 and Appendix 4, respectively. Appendix 5 contains an input data file to PREMOS for an example studied in Chapter 7.

CHAPTER 2

Large-Scale Circuit Simulation

2.1. Introduction

The simulation of LSI and VLSI circuits in their entirety falls beyond the capability of conventional circuit simulators. On the other hand, conventional logic simulators can only give the results in terms of logic levels "1", "0" or "unknown" with the attendant loss of detail in the waveforms. In recent years, many techniques have been proposed to bridge the gap between circuit simulation and logic simulation. The aim is to obtain a circuit-level type simulation with computational speeds approaching that of logic simulation. The analysis techniques used in 'conventional' circuit simulation and 'large-scale' circuit simulation are described in Sections 2.2 and 2.3, respectively. In Section 2.4, a discussion on problem areas with the previous methods is given.

2.2. Conventional Circuit Analysis

The equations that describe an integrated circuit model are generally a set of nonlinear (stiff) algebraic-differential equations of the following form[1]:

---

[1] In the sequel we use the lower case, such as x to denote a vector, x_i the ith element of x and upper case X a matrix.

$$f(x, \dot{x}, t) = 0, \quad x(0) = x_0 \tag{2.1}$$

Using an implicit integration formula, such as the backward Euler formula, the trapezoidal rule, or one of Gear's formula, (2.1) is approximated at every time point t_n by a set of nonlinear algebraic equations of the form:

$$g(x^n) = 0 \tag{2.2}$$

Eq. (2.2) is usually solved by using a modified Newton's method. At every iteration in the Newton's method, the linearized equations that have to be solved are of the form:

$$A\,x = b \tag{2.3}$$

A number of iterations may be necessary before the process converges to a solution of (2.2). At every iteration, function and Jacobian evaluations to construct the matrix A in (2.3), as well as LU decomposition and solution, are repeated. In practice, the matrix A is sparse and sparse matrix solution techniques can be used to reduce the computational requirements. The fundamental algorithm of circuit analysis can be summarized as follows:

```
BEGIN
  BEGIN
    X = [ Voltages, Currents]
    TIME = Start Time
    H = Initial Timestep
  END {initialization}
  TIME = TIME + H
  WHILE (TIME < End Time) DO
    BEGIN
      Discretize the differential operators by using an
      integration formula
```

```
      REPEAT
        BEGIN
          k = 1
          Evaluate linear models for circuit elements at the
          operating points and form the circuit matrix A
          and the source vector b
          Solve linear equations AX = b
        END
      UNTIL (convergence achieved) {dc loop}
      IF the local truncation error (LTE) is smaller than
      the tolerance
        THEN
          BEGIN
            Compute new timestep H
            TIME = TIME + H
          END
        ELSE
          BEGIN
            TIME = TIME - H
            Compute revised timestep H
            TIME = TIME + H
          END
    END {time loop}
END
```

For large-scale circuits, sparse matrix techniques alone do not produce simulation results in a reasonable time. To improve the speed of computation, tearing or decomposition together with latency detection and exploitation are used [12, 13, 14]. Depending on the computer algorithm implemented and on the circuit being analyzed, decomposition and latency checking could reduce the amount of computation two to five times [14]. In order to gain more computational speed, additional algorithms are needed, as described in the next section.

2.3. Large-Scale Circuit Analysis

The basic idea in many large-scale circuit simulators is, firstly, the partitioning of the circuit into smaller subcircuits, and then, the analysis of these subcircuits in a certain sequence. A number of algorithms have been proposed for the simulation of partitioned circuits using analysis sequencing.

2.3.1. Point Gauss-Jacobi Algorithm

In this algorithm, the components of x in (2.3) are obtained one at a time by solving a sequence of scalar equations; i.e., at time t_{n+1}, the kth component of x^{n+1}, x_k^{n+1}, is found by solving the scalar equation:

$$g_k\,(x_1^n,\ x_2^n,\ \ldots\ ,\ x_{k-1}^n,\ x_k,\ x_{k+1}^n,\ \ldots\ ,\ x_m^n) = 0 \qquad (2.4)$$

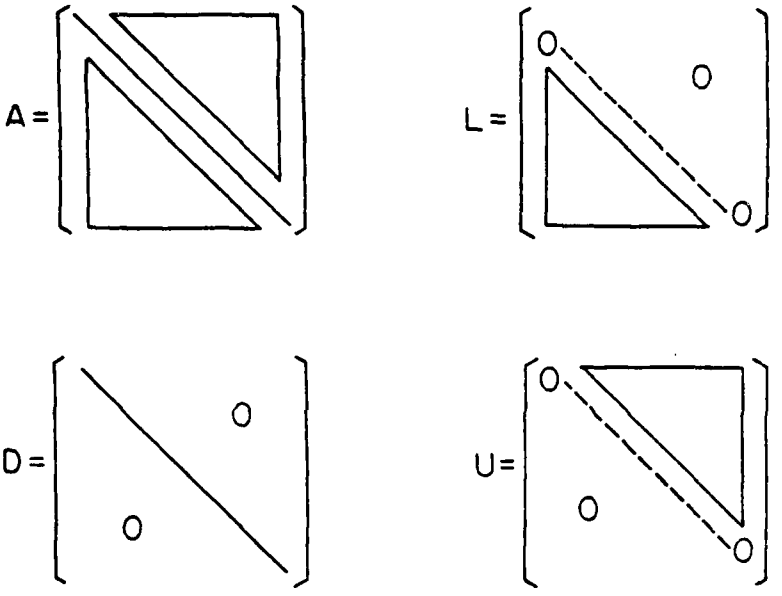
Eq. (2.4) can then be solved using a Newton method. In MOTIS and MOTIS-C [6, 7], a one-step regula-falsi iteration is used [16].

To illustrate this algorithm further, we partition the matrix A in (2.3) into the form

$$A = L + D + U \qquad (2.5)$$

where L and U are strictly lower and strictly upper triangular matrices and D is a diagonal matrix, as shown in Fig. 2.1. L_i and U_i stand for the ith row of the triangular matrices L and U. The point Gauss-Jacobi algorithm is described as

```
BEGIN
  BEGIN
    X = [ Voltages ]
    TIME = Start Time
```



$$A = L + D + U$$

Fig. 2.1 A = L + D + U.

```
    H = Initial Timestep
  END {initialization}
  TIME = TIME + H
  n = 1
  WHILE (TIME < End Time) DO
    BEGIN
      Discretize the differential operators by using an
      integration formula
      FOR node i, i=1 TO m DO
        BEGIN
          Evaluate linear models for nonlinear devices which
          are fanouts of ith node
          Form the row circuit matrix D_ii, L_i and
          U_i, and the current source b_i
          Solve linear equation

              D_ii X_i = b_i - L_i X_i^n - U_i X_i^n

        END {sweep m nodes}
      Compute new timestep H
      TIME = TIME + H
      n = n + 1
    END {time loop}
END
```

Since most automatic timestep control schemes are expensive for large-scale circuit simulation, fixed timestep during analysis has been used in some simulators like MOTIS and MOTIS-C.

2.3.2. Point Gauss-Seidel Algorithm

In this algorithm, the Gauss-Seidel technique is used to solve (2.3). At every iteration, one solves a sequence of scalar equations of the form:

$$g_k\,(x_1^{n+1},\ x_2^{n+1},\ \ldots,\ x_{k-1}^{n+1},\ x_k,\ x_{k+1}^{n},\ \ldots,\ x_m^{n}) = 0 \qquad (2.6)$$

The above equation could be solved by using Newton's method. In SPLICE [15] only one iteration is made.

Using the same expressions L_i, U_i and D_{ii} as in the last section, point Gauss-Seidel algorithm could be described as follows :

```
BEGIN
  BEGIN
    X = [ Voltages ]
    TIME = Start Time
    H = Initial Timestep
  END {initialization}
  TIME = TIME + H
  n = 1
  WHILE (TIME < End Time) DO
    BEGIN
      Discretize the differential operators by using an
      integration formula
      FOR node i, i=1 TO m DO
        BEGIN
          Evaluate linear models for nonlinear devices which
          are fanouts of ith node
          Form the row circuit matrix D_ii, L_i and
          U_i, and the current source b_i
          Solve linear equation

            D_ii X_i = b_i - L_i X_i^(n+1) - U_i X_i^n

        END {sweep m nodes}
      Compute new timestep H
      TIME = TIME + H
      n = n + 1
    END {time loop}
END
```

2.3.3. Block Gauss-Seidel-Newton Algorithm

From the network point of view, the point Gauss-Jacobi and the point Gauss-Seidel methods are equivalent to decomposing the network at every node. In the block Gauss-Seidel-Newton algorithm, the network is decomposed into subcircuits, which may consist of more than one node. A Gauss-Seidel-Newton method is then used to solve the partitioned system of equations, which now becomes a sequence of subcircuit equations, rather than scalar equations, of the form:

$$g_k\ (x_1^{n+1},\ x_2^{n+1},\ \ldots\ ,\ x_{k-1}^{n+1},\ x_k,\ x_{k+1}^{n},\ \ldots\ ,\ x_m^{n}) = 0 \qquad (2.7)$$

where the x_i is now a vector. Note that a modified Newton-Raphson method could be used in solving each subcircuit.

In this algorithm, the matrix A is partitioned into the form

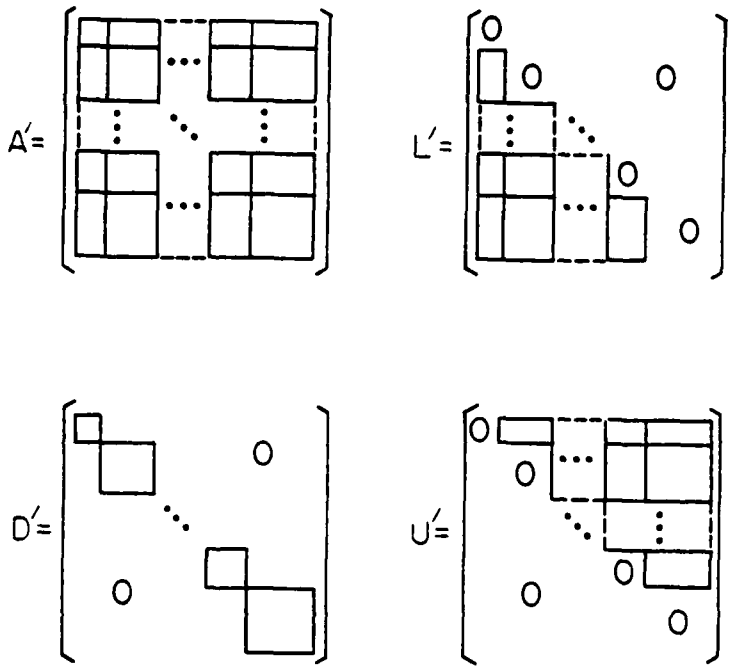
$$A = L' + D' + U' \qquad (2.8)$$

where L' and U' are strictly lower block and strictly upper block triangular matrices and D' is the block diagonal matrix (Fig. 2.2). Here, L_i' and U_i' represent the ith block matrices of the L' and U', respectively. D_{ii}' is the ith block diagonal matrix. The block Gauss-Seidel-Newton algorithm can be described as follows :

```
BEGIN
  BEGIN
    X = [ Voltages ]
    TIME = Start Time
    H = Initial Timestep
  END {initialization}
  TIME = TIME + H
  n = 1
  WHILE (TIME < End Time) DO
    BEGIN
      Discretize the differential operators by using an
      integration formula
      FOR subcircuit i, i=1 TO m DO
        BEGIN
          REPEAT
            BEGIN
               Evaluate linear models for nonlinear devices of
               ith subcircuit
               Form the block circuit matrix D'ii, L'i and
               U'i, and the current source vector b'i
               Solve linear equations

                 D'ii Xi = b'i - L'i X(n+1)i - U'i Xni

            END
          UNTIL (nonlinear converged) {dc loop for kth subcircuit}
        END {sweep m subcircuits}
      Compute new timestep H
```



$$A = L' + D' + U'$$

Fig. 2.2 $A = L' + D' + U'$.

```
      TIME = TIME + H
      n = n + 1
    END {time loop}
END
```

2.3.4. Waveform Relaxation Method

For the three algorithms described above, the circuit analysis proceeds by small timesteps at the global level, as is done in conventional circuit simulation. In the waveform relaxation method, the waveforms are obtained for a time interval at the subcircuit level and a number of waveform iterations are then taken to converge to the solution [17]. Either the Gauss-Jacobi method or the Gauss-Seidel method could be used in the waveform relaxation algorithm. The Gauss-Seidel waveform relaxation algorithm can be described as follows:

```
BEGIN
  X = [ Voltages, Currents ]
  n = 1
  WHILE (δ^n < Tolerance) DO
    BEGIN
      FOR subcircuit i, i=1 TO m DO
        BEGIN
          FOR time t=0 TO t=End Time DO
            BEGIN
              Solve nonlinear equations
                .
                X_i^{n+1} = f_i(X_1^{n+1},...,X_{i-1}^{n+1},X_i^{n+1},X_{i+1}^n,...,X_m^n)
                  and  X_i^{n+1}(0) = X_i^n(0)
            END
        END {sweep m subcircuits}
      δ^{n+1} = max max | X^{n+1}(t) - X^n(t) |
                 i   t
      n = n + 1
    END {waveform iteration loop}
END
```

The waveform relaxation method is attractive since the subcircuits are analyzed independently and thus different timesteps could be used. However, the method suffers from convergence problems when strong feedback exists. Furthermore, a number of iterations are needed for the waveforms to converge to the solution and a large memory is required to store the entire waveforms at each iteration.

2.4. Problems with The Previous Methods

In general, the large-scale circuit simulation algorithms described above have the following features in common:

(1) decomposing the entire circuit into small subcircuits and adopting the circuit analysis for each subcircuit sequentially

(2) using relaxation methods in solving the circuits

(3) using simplified device models for circuit elements.

It is obvious that there are some tradeoffs between the speed of simulation and the accuracy of the simulated results, which depend upon the accuracy requirement. The problems and the impacts for large-scale circuit simulation are discussed in the following :

(i) circuit decomposition

As the size of circuit increases, the time required to solve the circuit equations increases very fast and rapidly becomes the dominant cost of the analysis in conventional circuit analysis [15]. Decomposing the circuit into small subcircuits and analyzing the circuit at subcircuit level reduces the computation time because it now

grows almost linearly with circuit size. As the average size of the subcircuits is reduced, the growth in total computation time becomes more linear. For example, the subcircuit size in [6, 7] is always one. However, if the size of each subcircuit is forced to be one, then the interactions among the subcircuits may be strong, which could affect the accuracy of the one-iteration Gauss-Jacobi or Gauss-Seidel approach. This problem is discussed in Chapter 4.

(ii) device modeling

There are generally two forms for representing device characteristic models: functional form and tabular form. The former is generally used in circuit simulation, where nonlinear model equations and parameters are employed to describe the operations of the devices. The latter is often used in timing simulation and piecewise-linear analysis methods. The tabular models could be in one dimensional or two dimensional form [6, 7, 8, 18]. Depending on different requirements, either one or a combination of these two approaches can be used for device modeling. In Appendix 1, both MOS device modeling and capacitor modeling for VLSI circuits are discussed.

(iii) nonlinear iterations

Most circuit simulation programs use the (modified) Newton-Raphson algorithm to determine the solution of nonlinear system of algebraic equations. The criterion for the convergence of the iterative solutions is the requirement that the vector of circuit variables x_{k+1} agrees with the prior solution x_k within a specified tolerance. For large-scale circuit analysis, it is rather expensive

to use the same convergence criterion in conventional circuit simulation. In [6, 7], only one iteration is taken at each timepoint. In [8], a relaxation method is used to reduce the number of nonlinear iterations. More on this topic is discussed in Chapter 4.

(iv) timestep control scheme

In conventional circuit simulation, the timestep is usually controlled by the local truncation error (LTE) [5]. For large-scale circuit simulation, it is too costly to use the LTE timestep control scheme. Some of the existing techniques use a fixed timestep scheme [6, 7]. A simple variable timestep control scheme, where the internal timestep changes according to circuit activity, is adopted in [18]. In [8], an iteration count timestep control scheme is used. In Chapter 6, more exploitation of the timestep control scheme is done.

(v) accuracy

From the accuracy point of view, it has been found that both the Gauss-Jacobi and the Gauss-Seidel methods tend to produce a response that lags behind the actual response. In Chapter 5, a modified Gauss-Seidel method is described to solve the partitioned system of equations. It is shown, by examples, that the new method is more accurate than the standard Gauss-Seidel method. The proposed modified Gauss-Seidel method is proved to be consistent, stable and convergent.

CHAPTER 3

Analysis Sequencing

3.1. Introduction

In order to analyze a large-scale circuit, the entire circuit is usually partitioned into smaller 'one-way' subcircuits at first, and then, these subcircuits are analyzed in a certain sequence [9]. To create 'one-way' subcircuits requires, in general, the introduction of some approximations. For MOS circuits, 'one-way' subcircuits are created by decoupling the gate-to-drain capacitance. To allow the subcircuits to be analyzed independently in sequence, a scheduling scheme is followed. This scheduling process is called analysis sequencing.

By properly defining the subcircuits in combinational logic circuit, the overall circuit equations can be ordered into a lower block-triangular form (LBT) [12], so that analysis sequencing can be applied most efficiently. In general, when there is feedback among the subcircuits, such as in sequential circuits, the circuit equations cannot be ordered into a lower block-triangular form unless the sizes of the subcircuits are increased to include the feedback [9]. Alternatively, the (block) Gauss-Jacobi or (block) Gauss-Seidel technique may be used to decouple the equations for analysis sequencing and to keep the sizes of the subcircuits relatively small. This decoupling in effect 'breaks' the feedback in the analysis sequencing

procedure.

In Section 3.2, some mathematical properties on directed graphs are discussed. Section 3.3 summarizes previous work on 'levelizing' the vertices in an acyclic directed graph. Two new algorithms based on programming data structures are developed in this thesis and described in Section 3.4; one algorithm uses a stack and the other a queue. Examples are shown to compare the differences between these two algorithms. Since in many cases, feedback may exist in the network, an algorithm for checking feedback paths is studied in Section 3.5. In addition to latency and selective trace (or event driven), an important idea for further saving of CPU time and memory is to schedule only those subcircuits that directly or indirectly affect the outputs in the circuit analysis. This concept of scheduling only 'relevant' subcircuits together with a corresponding algorithm are described in detail in Section 3.6. An algorithm for analysis sequencing using parallel processing is proposed in Section 3.7. The last section, Section 3.8, gives the conclusions.

3.2. Mathematical Properties

A circuit which is composed of unilateral subcircuits can be represented by a directed graph G(V,E), where each vertex in V corresponds to each subcircuit and each edge in E corresponds to each signal line from fanout to fanin. The mathematical properties of directed graphs can be found in many books on graph theory, such as

[19, 20, 21]. To simplify the description of the scheduling algorithms proposed in this thesis, the following definitions and derivations are given :

Definition 3.1 :

Given a vertex v of G(V,E), the set of fanin vertices and fanout vertices of v are defined as

$$\begin{aligned} fin(v) &= \{ w \in V \mid (w,v) \in E \} \\ fout(v) &= \{ w \in V \mid (v,w) \in E \} \end{aligned} \tag{3.1}$$

The number of fanin and fanout vertices of v are defined as nfin(v) and nfout(v), respectively.

The adjacency matrix $X = [x_{ij}]$ of the directed graph G(V,E) is defined as a n by n matrix whose element

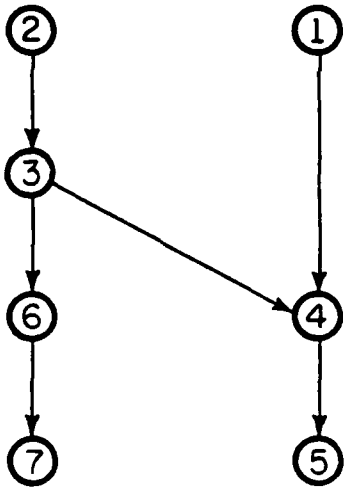
$x_{ij}=1$ if there is an edge directed from ith vertex to jth vertex
$=0$ otherwise.

A directed graph and its adjacency matrix are shown in Fig. 3.1. It is easy to observe the following two properties :

1. nfin(v) is the sum of the column of corresponding vertex v.

2. nfout(v) is the sum of the row of corresponding vertex v.

Note that any set of parallel directed edges in G(V,E) will be treated as one edge, without affecting the analysis sequencing. If X is the adjacency matrix of G(V,E), then the transposed matrix X^T is the adjacency matrix of a directed graph $G^R(V,E)$ obtained by reversing the direction of every edge in G(V,E). The following relation can be derived.



(a)

$$X = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & & & 1 & & & \\ & 0 & 1 & & & & \\ & & 0 & 1 & & 1 & \\ & & & 0 & 1 & & \\ & & & & 0 & & \\ & & & & & 0 & 1 \\ & & & & & & 0 \end{bmatrix} \end{array}$$

(b)

Fig. 3.1 (a) A Directed Graph.
(b) The Adjacency Matrix X.

Lemma 3.1 [19] :

nfin(v) in $G^R(V,E)$ equals nfout(v) in G(V,E) and nfout(v) in $G^R(V,E)$ equals nfin(v) in G(V,E).

For the directed graph G(V,E) with no feedback loops (i.e., G(V,E) is acyclic), the analysis sequencing is to reorder the rows (columns) in the corresponding adjacency matrix X and make the matrix X upper triangular. For illustration, the following definitions are made.

Definition 3.2 :

Vertex v_i in G(V,E) is a predecessor of vertex v_j if and only if there is a directed path from v_i to v_j. If v_i is a predecessor of v_j, then v_j is a sucessor of v_i.

Definition 3.3 :

A linear ordering is called a topological order if it has the property that if v_i is a predecessor of v_j in the network, then v_i precedes v_j in the linear ordering.

For an acyclic directed graph, the analysis sequencing is to arrange the vertices in a topological order and the following theorem is obtained.

Theorem 3.1 [19] :

The vertices in a directed graph can be arranged in a topological order if and only if the directed graph is acyclic.

In general, when feedback loops exist in the directed graph, the graph is no longer acyclic. In this case, analysis sequencing procedures need to be extended to check for the feedback paths and to schedule the analysis of the subcircuits in the proper sequence.

3.3. Previous Work

Over the past few years, two methods have been proposed for sequencing the vertices of general directed graphs which are not necessarily acyclic. One method is to construct a new acyclic directed graph G' first by contracting the vertices in each strongly connected component of the original graph G into a new vertex in G' [9]. Tarjan's algorithm [22] could be used to find the strongly connected components of G in linear time complexity. The vertices in G' are then levelized and scheduled by Algorithm 3.1 given below.

Algorithm 3.1 [9]

The notation $nu(v_i)$ is the updated number of fanin vertices of v_i after all the scheduled vertices have been removed.

```
Part I : (assignment of vertices of G'(V,E) to levels)
  BEGIN
     Assign input vertices of G'(V,E) to level 0;
     k ← 0;
L.   FOR each vertex v in level k DO
        FOR each vertex w ∈ fout(v) DO
           BEGIN
```

```
            nfin(w) ← nfin(w)-1
            IF nfin(w)=0 THEN
            assign w to level k+1;
          END
        IF level k is not empty THEN
        GO TO L;
        k ← k+1
    END
Part II : (scheduling the analysis of the subcircuits)
  BEGIN
     k ← 1;
L.   FOR each vertex v of G'(V,E) at level k DO
        time analysis of corresponding subcircuits;
        IF level k is non-empty THEN
        GO TO L;
        k ← k+1;
  END
```

Algorithm 3.1 is illustrated by the following example :

Example 3.1 :

For the directed graph G'(V,E) shown in Fig. 3.2, Algorithm 3.1 levelized it with depth 8 :

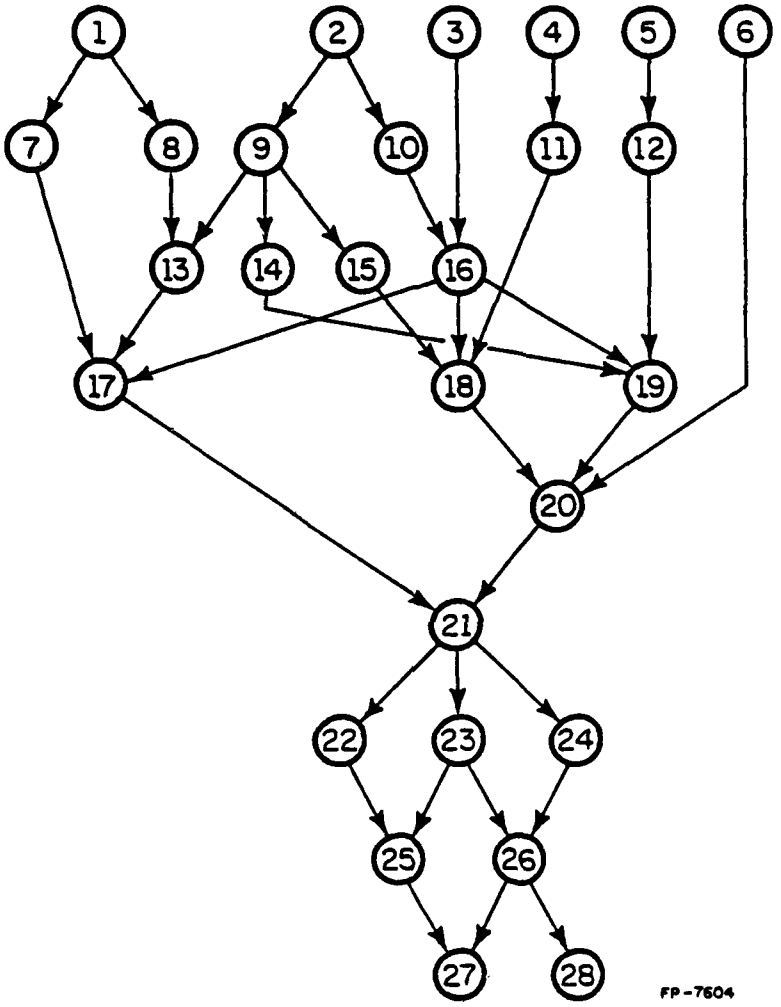


Fig. 3.2 Directed Graph G'(V,E).

Level	0	1	2	3	4	5	6	7	8
	1	7	13	17	20	21	22	25	27
	2	8	14	18			23	26	28
	3	9	15	19			24		
	4	10	16						
	5	11							
	6	12							

Remark 3.1 :

The principle of levelizing the vertices is that a vertex v is in level k if all the vertices of fin(v) belong to levels numbered from 0 to k-1. The depth of an acyclic directed graph is the maximum value of levels [23].

Remark 3.2 :

In large-scale circuit analysis, one should try to keep the size of the subcircuits small in order to make the total analysis time linearly proportional to the size of the entire circuit. However, the sizes of the subcircuits (or vertices) after contraction in Algorithm 3.1 could become too large for the analysis to be efficient. For example, a N-stage ring oscillator would be contracted into one subcircuit instead of N subcircuits.

Another method [8] deals with the directed graph G(V,E) directly without contracting it as is done in Algorithm 3.1. When a feedback loop is found, the loop is broken. Levelizing the vertices in this way is performed by using the following algorithm.

Algorithm 3.2 [8]

For describing the algorithm, the following notations are

defined :

adj(v) : set of adjacent vertices corresponding to the set of the incoming edges of vertex v.

la(v) : label of vertex v.

Procedure :

1. Set $la(v_i)=0$ for each vertex v_i of G(V,E)
2. $la(v_i)=1$ for each vertex v_i which corresponds to an input signal terminal.

   k=1.
3. k=k+1.

   Choose a vertex v_j where $la(v_j)=0$ and $la(v_i)\neq 0$ for all $v_i \in adj(v_j)$. If there is no such vertex, choose a vertex v_j connecting to a vertex which has the lowest label. $la(v_j)=k$.
4. Repeat step (3) until all the vertices in G(V,E) are labeled.

Remark 3.3 :

The level in Algorithm 3.1 is equivalent to the label in Algorithm 3.2, except the label in Algorithm 3.2 starts from one instead of zero.

Remark 3.4 :

It is claimed in [8] that Algorithm 3.2 can find all feedback loops which will be cut during the analysis sequencing. But, as shown in the following example, Algorithm 3.2 sometimes fails in identifying the proper feedback loops.

Example 3.2 :

For the directed graph shown in Fig. 3.3, it is obvious that (7,3) is the feedback path that should be identified. The correct sequence for this directed graph is as follows :

Label :	1	2	3	4
	1	3	4	5
	2		6	7

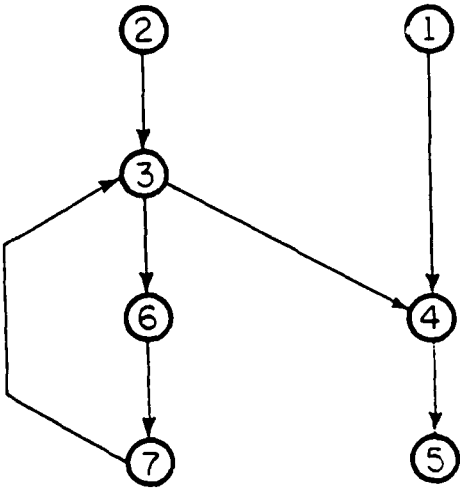
But by using Algorithm 3.2, the following sequence could be produced :

Label :	1	2	3	4	5	6
	1	4	5	3	6	7
	2					

In the next section, a more accurate algorithm for identifying the feedback paths will be given.

3.4. New Algorithms Suitable for Computer Implementation

In the computer implementation of scheduling algorithms data structuring is important, particularly when multi-processor computer configurations are to be taken into consideration. In this section, two scheduling algorithms based on two different data structures are described. The first is based on a stack and the second on a queue. For a single processor computer, both algorithms will have the same performance since only one task can be carried out at any given time. However, for a multi-processor computer, the second algorithm, whose data structure is based on a queue, is more efficient than the first



FP-6729

Fig. 3.3 A Directed Graph.

algorithm.

Definition 3.4 :

A stack is a linear list for which all insertions and deletions (and usually all accesses) are made at one end of the list. So the stack is a last-in-first-out ("LIFO") list [24].

Definition 3.5 :

A queue is a linear list for which all insertions are made at one end of the list; all deletions (and usually all accesses) are made at the other end. So the queue is a first-in-first-out ("FIFO") list [24].

In [25], a topological sorting algorithm based on stack is proposed. The computing time is O(|V|+|E|), which is linear in the size of the problem. The following algorithm performs analysis sequencing using the stack data structure.

Algorithm 3.3

The notation s(i), i=1,m, is the sequence of analyzing vertices v_j, j=1,m, for the entire directed graph G(V,E).

```
BEGIN
  k=1
  FOR each vertex v_i in G(V,E) DO
    BEGIN
      nu(v_i)=nfin(v_i)
      IF nu(v_i)=0
        THEN push the vertex v_i into the stack S
    END
  REPEAT
    IF the stack S is not empty
      THEN
        BEGIN
          pop out v_j from the stack S
          s(k)=v_j
          k=k+1
          FOR each vertex v_i in fout(v_j) DO
            BEGIN
              nu(v_i)=nu(v_i)-1
              IF nu(v_i)=0
                THEN push v_i into the stack S
            END
        END
      ELSE
        BEGIN
          check the feedback path
          (see the Algorithm 3.5)
          push the associated vertex into the stack S
        END
  UNTIL all vertices in G(V,E) are scheduled (k>m)
END
```

Example 3.3 :

For the directed graph in Fig. 3.2, Algorithm 3.3 gives the

following sequence :

6 5 12 4 11 3 2 10 16 9 15 18 14 19
20 1 8 13 7 17 21 24 23 26 28 22 25 27

Fig. 3.4 shows the flow of this sequence.

Algorithm 3.4 is similiar to Algorithm 3.3 except that Algorithm 3.4 is based on the queue instead of the stack. In the implementation, a circular queue is used to prevent memory overrun.

Algorithm 3.4

```
BEGIN
  k=1
  FOR each vertex v_i in G(V,E) DO
    BEGIN
      nu(v_i)=nfin(v_i)
      IF nu(v_i)=0
        THEN push the vertex v_i into the queue Q
    END
  REPEAT
    IF the queue Q is not empty
      THEN
        BEGIN
          pop out v_j from the queue Q
          s(k)=v_j
          k=k+1
          FOR each vertex v_i in fout(v_j) DO
            BEGIN
              nu(v_i)=nu(v_i)-1
              IF nu(v_i)=0
                THEN push v_i into the queue Q
            END
        END
      ELSE
        BEGIN
          check the feedback path
          (see the Algorithm 3.5)
          push the associated vertex into the queue Q
        END
  UNTIL all vertices in G(V,E) are scheduled (k>m)
END
```

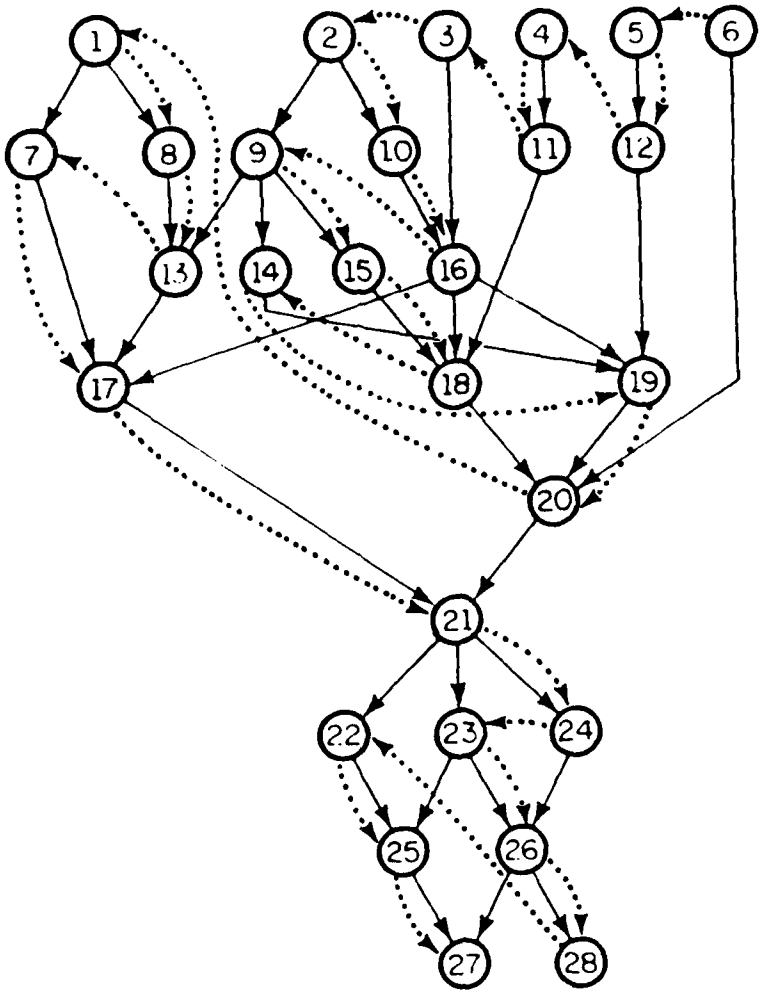


Fig. 3.4 The Sequence Given by Algorithm 3.3 for the Directed Graph in Fig. 3.2.

Algorithm 3.4 is illustrated by Example 3.4.

Example 3.4 :

For the directed graph in Fig. 3.2, Algorithm 3.4 gives the following sequence :

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28

The flow diagram of this sequence is shown in Fig. 3.5.

Remark 3.5 :

Comparing the sequences shown in Fig. 3.4 and 3.5, it can be seen that the algorithm based on the stack generates the sequence by using depth-first search, while the algorithm based on the queue constructs the sequence by selecting all the vertices at one level and advancing level by level.

3.5. Discussion on Checking Feedback Path

For the acyclic directed graph, the vertices can be arranged in a topological order by any analysis sequencing procedure. But it is possible that there exist feedback loops in many networks. As stated in Section 3.3, the feedback loops can be avoided by contracting the strongly connected component into a new vertex and the new constructed directed graph is then acyclic [9]. In some cases, this approach may not be efficient since the new generated subcircuit could be very large. For solving large-scale networks, the Gauss-

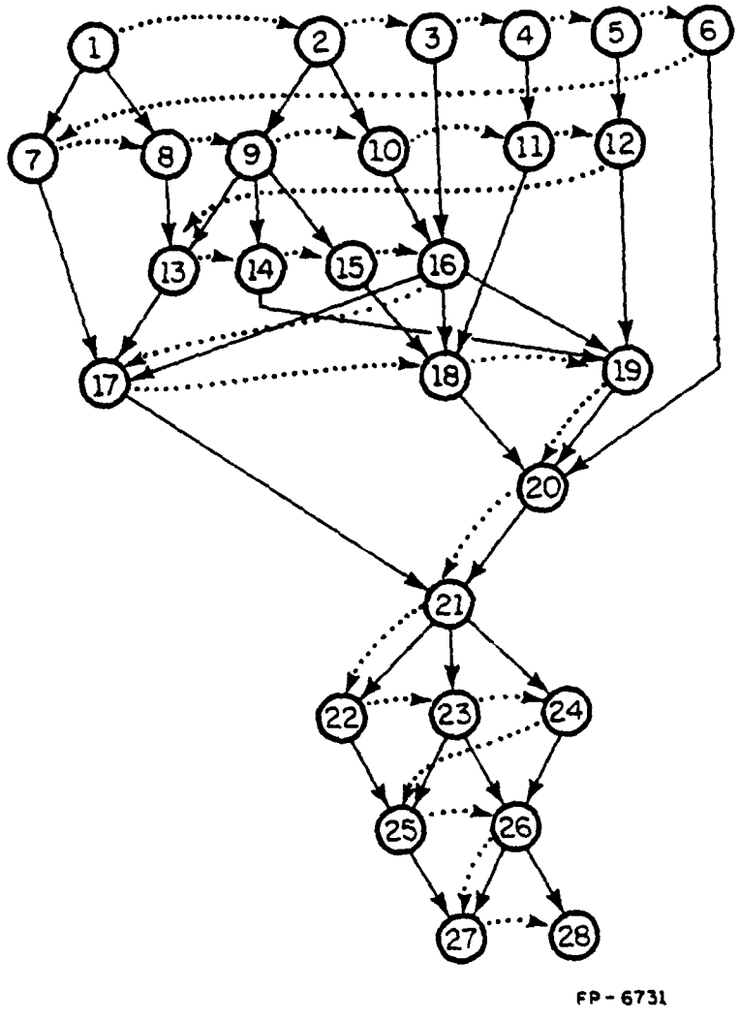


Fig. 3.5 The Sequence Given by Algorithm 3.4 for the Directed Graph in Fig. 3.2.

Seidel technique is widely used; this technique is equivalent to breaking the feedback paths [8, 15]. Thus it is necessary to check and identify the feedback paths in the graph. Furthermore, as the predictor method [see Chapter 5] will be adopted to predict the voltage on the feedback loop, it is necessary to store information concerning the feedback checking.

In Algorithm 3.3 (Algorithm 3.4), the stack (the queue) stores the 'unscheduled' vertices whose fanin vertices have been scheduled. Thus the vertices in stack (queue) are ready tasks for sequencing [26]. If there are feedback loops in the network, an empty stack or empty queue would result before all the vertices in the analysis sequencing procedures have been scheduled. It is assumed that all feedback paths are single; i.e., that only one feedback path enters a vertex, and that no feedback enters the input vertices (in general, input vertices correspond to independent sources). It is straightforward to obtain the following lemma.

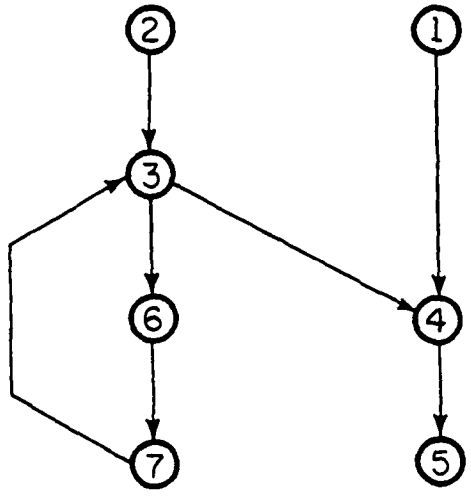
Lemma 3.2 :

In a directed graph G(V,E), a vertex v with single feedback path entering it is identified if

(1) nu(v) = 1 at the point of sequencing, and

(2) nfin(v) > 1.

Fig. 3.6 shows a directed graph with a single feedback path and its adjacency matrix X. After the input vertices v_1 and v_2 are scheduled, the associated rows and columns of v_1 and v_2 are removed



(a)

$$X = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & & & 1 & & & \\ & 0 & 1 & & & & \\ & & 0 & 1 & & 1 & \\ & & & 0 & 1 & & \\ & & & & 0 & & \\ & & & & & 0 & 1 \\ & & & & & & 0 \end{bmatrix} \end{array}$$

(b)

Fig. 3.6 (a) A Directed Graph with a Single Feedback Path.
(b) The Adjacency Matrix X.

from X. The modified matrix becomes

$$X' = \begin{array}{c} \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccc} 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 1 & & 1 & \\ & 0 & 1 & & \\ & & 0 & & \\ & & & 0 & 1 \\ 1 & & & & 0 \end{bmatrix} \end{array}$$

It is clear that the sum of each column equals to the number of fanins at this point; that is $nu(v_3)=nu(v_4)=nu(v_5)=nu(v_6)=1$. By applying the above Lemma, we have two candidates, v_3 and v_4. Both have the same number of original fanins as two. Which one should be scheduled next? Obviously v_3 is the answer. In the following, a new algorithm for checking feedback path is listed :

Algorithm 3.5

In Algorithm 3.5, a depth-first search is carried out for finding the single feedback path (v',v) from v' to v. The stack or queue is empty at the outset.

The notation $lab(v_i)$ is defined as

$lab(v_i)=1$ if v_i is visited

$=0$ elsewhere

```
BEGIN
  search the set of vertices V_o, where for each vertex
   v in V_o nu(v)=1 and nfin(v)>1.
L choose a vertex v_i in V_o with lab(v_i)=0
    BEGIN
      FOR each vertex v_j in fout(v_i) DO
        BEGIN
          IF lab(v_j)=0
            THEN
```

```
                BEGIN
                  lab(v_j)=1
                  push v_j into the queue Q"
                END
          END
L"      pop out the vertex v_j from the queue Q"
        FOR each vertex v_k in fout(v_j) DO
          BEGIN
            IF v_k=v_i
              THEN GO TO L' {feedback path is found}
              ELSE
                BEGIN
                  IF lab(v_k)=0
                    THEN
                      BEGIN
                        lab(v_k)=1
                        push v_k into the queue Q"
                      END
                END
          END
        IF the queue Q" is NOT empty
          THEN GO TO L"
          ELSE GO TO L
      END {L}
L' BEGIN
      v=v_i
      v'=v_j
      (v',v) is the feedback path
    END
  END
```

Example 3.5 illustrates the procedures given in this algorithm.

Example 3.5 :

Consider the directed graph in Fig. 3.6, where a feedback path exists from vertex 7 to vertex 3. After vertex 1 and 2 are scheduled by using Algorithm 3.3 or Algorithm 3.4, vertex 3 and 4 are selected as candidates for the possibility of having feedback paths entering them. Following Algorithm 3.5, two possibilities exist :

(1) If vertex 4 is chosen first, then vertex 4 and 5 are marked and the search ends at vertex 5. Therefore, no feedback path enters

vertex 4. Vertex 3 is selected next; and the search proceeds through vertex 6 and 7 and terminates back at 3. Therefore, a feedback path exists from 7 to 3. Further search will bypass 4 since it has already been marked 'old'.

(2) If vertex 3 is chosen first, vertex 4, 6, 5 and 7 are visited and marked sequentially before the search terminates back at vertex 3, which again indicates a feedback path from 7 to 3.

Remark 3.6 :

Since each vertex is labeled at most once and each edge is examined at most once, the time complexity for this algorithm is $O(|V|+|E|)$.

Remark 3.7 :

If there is no feedback loop in the network, then Algorithm 3.5 would not take any computation time in checking feedback.

Remark 3.8 :

For arbitrary networks, this algorithm may not be satisfactory in identifying minimal feedback loops as other complex algorithms do [27]. However, general algorithms are not cost effective because the complexity grows exponentially with the size of the network [27]. Algorithm 3.5, on the other hand, is cost effective for digital electronic networks, where the feedback loops are generally regular and simple.

3.6. Analysis Sequencing for Relevant Parts

For most digital circuits, only a small portion of the entire circuit is active at any time. Latency and selective trace (or event scheduler) techniques have been used to take advantage of this fact and save CPU time and memory storage [13, 14]. Latency exploitation amounts to identifying the inactive parts of the circuit at each timepoint in the solution process and bypassing them at that timepoint. In contrast, the selective trace technique depends on finding the active parts and analyzing them in the proper sequence. Beyond these two techniques, a new technique which could save additional computation time and memory is described next.

In many cases, especially in digital circuits, the output of interest may be directly or indirectly affected by only a subset of the subcircuits in the system. These subcircuits will be referred to as the 'relevant' parts of the system. During the simulation, it is only necessary to analyze the relevant parts even if the remaining parts of the system are active. For example, Fig. 3.7 (a) shows the entire circuit to be analyzed, which has been partitioned into seven unilateral subcircuits. If one is only interested in the output of subcircuit 7, then, instead of all seven subcircuits, only four subcircuits 1, 2, 4 and 7 need to be scheduled and analyzed (Fig. 3.7 (b)). Similiar results can be obtained as shown in Fig. 3.7 (c) and (d) if the output of subcircuit 6 only, or subcircuit 5 only, are of interest. The concept of analyzing only 'relevant' parts is similiar to the concept of 'segementation' in logic simulation [27].

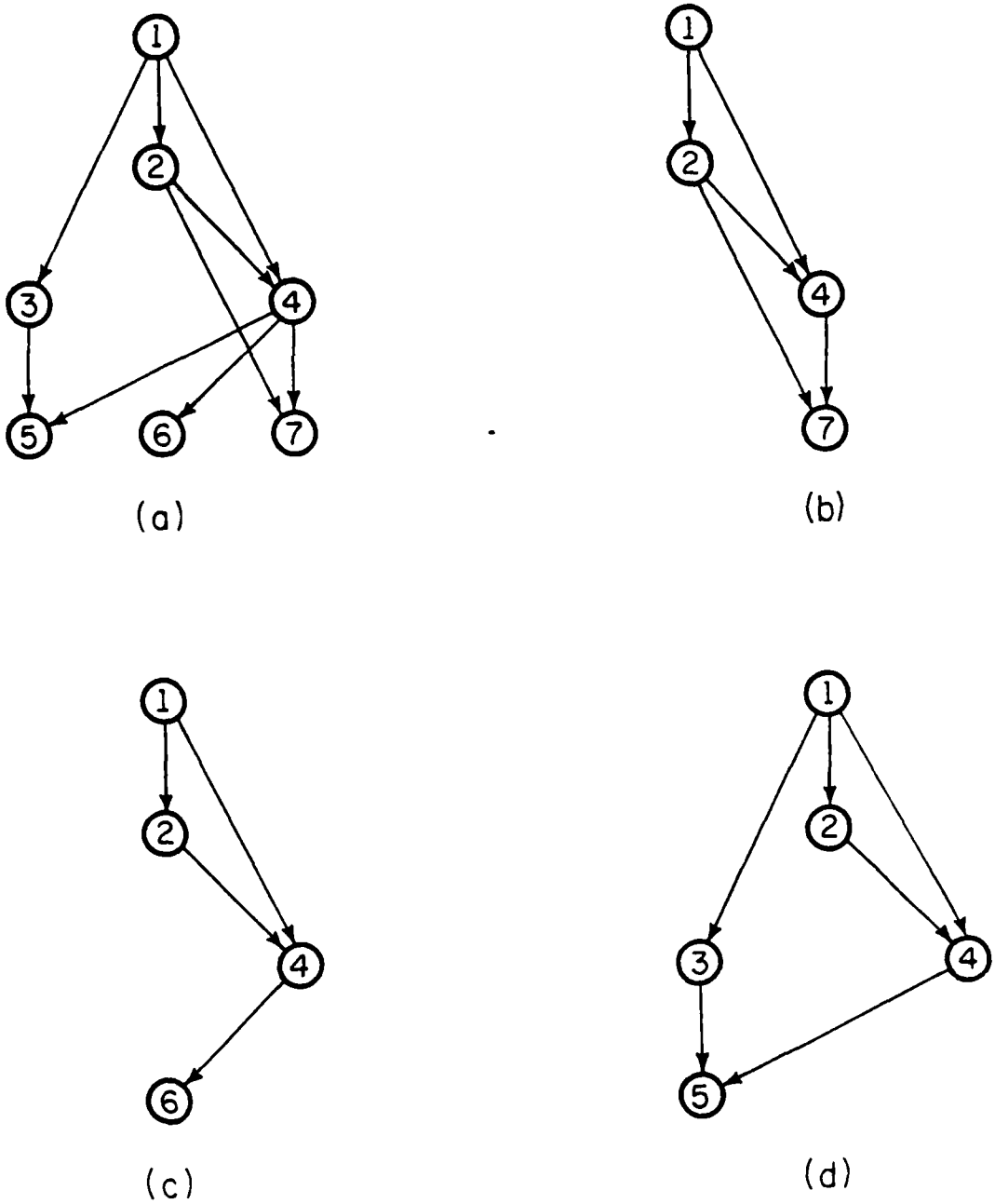


Fig. 3.7 An Illustrative Circuit.

This approach can be combined with either a latency technique or a selective trace technique to further increase the speed of simulation. For the large-scale circuit simulation, an efficient algorithm of scheduling the relevant parts is given below. First, the sequence of analysis is constructed for the G(V,E) corresponding to the entire circuit by using Algorithm 3.3 or Algorithm 3.4. Second, the vertices associated with the relevant parts are found and labeled by tracing backward from the outputs of interest to the inputs. Finally, the nonrelevant vertices are deleted from the sequence of analysis. Then the remaining sequence identifies the relevant parts.

Algorithm 3.6

This algorithm is to sequence the vertices of the relevant parts only.

To describe the algorithm, we use the following notations :

lcr(i), i=1,m', : the sequence of analyzing vertices v_j, j=1,m', of the relevant parts only.

lbk(v_i) = 1 if v_i is in the relevant set

= 0 elsewhere.

The algorithm is composed of three parts :

Part I : (analysis sequencing for the directed graph G(V,E) which corresponds to the entire circuit)

```
BEGIN
  k=1
  FOR each vertex v_i in G(V,E) DO
    BEGIN
      nu(v_i)=nfin(v_i)
      IF nu(v_i)=0
```

```
          THEN push the vertex v_i into the queue Q
      END
    REPEAT
      IF the queue Q is not empty
        THEN
          BEGIN
            pop out v_j from the queue Q
            s(k)=v_j
            k=k+1
            FOR each vertex v_i in fout(v_j) DO
              BEGIN
                nu(v_i)=nu(v_i)-1
                IF nu(v_i)=0
                  THEN push v_i into the queue Q
              END
          END
        ELSE
          BEGIN
            check the feedback path
            (see the Algorithm 3.5)
            push the associated vertex into the queue Q
          END
    UNTIL all vertices in G(V,E) are scheduled (k>m)
  END
```

Part II : (identifying the relevant vertices by tracing them backward from the vertices v_{oi}, i=1,m", whose voltage values are of interest)

Reverse every edge in G(V,E) and obtain $G^R(V,E)$. Since fin(v) and fout(v) in G(V,E) are fout(v) and fin(v) in $G^R(V,E)$, let $fin^R(v)=fout(v)$ and $fout^R(v)=fin(v)$.

```
  BEGIN
    FOR each vertex v_oi DO
      BEGIN
        push the vertex v_oi into the queue Q'
        lbk(v_oi)=1
      END
    REPEAT
      BEGIN
        pop out the vertex v_j from the queue Q'
        IF nfin(v_j)≠0
          THEN
            BEGIN
              FOR each vertex v_i in fout^R(v_j) DO
```

```
                BEGIN
                  IF lbk(v_i)=0
                    THEN
                      BEGIN
                        push v_i into the queue Q'
                        lbk(v_i)=1
                      END
                END
            END
        END
      UNTIL the queue Q' is empty
    END
```

Part III : (deleting nonrelevant vertices from the sequence obtained in Part I and giving the sequence of the relevant set only)

```
    BEGIN
      k=0
      FOR i=1 TO m  DO
        BEGIN
          IF lbk(s(i))=1
            THEN
              BEGIN
                k=k+1
                lcr(k)=s(i)
              END
        END
    END
```

3.7. Extension to Multiprocessor Computer

First, we assume that the analysis time for each vertex is the same and that the directed graph is acyclic. If there is no limitation on the number of processors or if the number of processors available is not less than the maximum number of processors required for each level of the sequence, then the minimum computation time is obtained.

Lemma 3.3 :

If an unlimited number of processors are available or if the number of processors available is not less than the maximum number of processors required, then the minimum completion time of the solution process is the number of levels, which is the length of longest path in the directed graph.

For example, for the directed graph in Fig. 3.2, the minimum completion time is 9.

When the number of processors required is too large, many of them will be idle most of the time, which is not economical. In practice, there is always a limit on the number of processors available in a multi-processor computer system. Therefore, we will consider next the case when a limited number of processors are available.

In recent years, many scheduling strategies have been proposed to process the task directed graph with the number of processors available [28]. These strategies show different levels of complexity and give different degrees of processor utilization.

Reverse each edge in G(V,E) and obtain the reversed graph $G^R(V,E)$. The level number assigned by Algorithm 3.1 to a vertex v in $G^R(V,E)$ is $lev^R(v)$.

Definition 3.6 :

The rank r(v) of a vertex v is defined to be

$$r(v) = D - lev^R(v)$$

where D is the depth of the directed graph G(V,E).

Example 3.6 :

The rank sequence for the directed graph in Fig. 3.2 is

rank	0	1	2	3	4	5	6	7	8
	2	1	8	6	17	21	22	25	27
		3	11	7	20		23	26	28
		4	12	13			24		
		5	14	18					
		9	15	19					
		10	16						

The rank number is the latest time that a vertex must be scheduled to have the minimum completion time for the directed graph.

A ready task has been defined to be the vertex v for which the vertices in fin(v) have all been scheduled. The strategy followed is to schedule the vertex with the smallest rank number among all the ready tasks.

Example 3.7 :

For the directed graph shown in Fig. 3.2, if the number of processors available is p=3, the sequences by using the above stategy are

p=3 sequence	1	2	3	4	5	6	7	8	9	10	11
	2	4	10	12	16	13	17	21	22	25	27
	1	5	8	14	6	18	20		23	26	28
	3	9	11	15	7	19			24		

If the strategy is to schedule the one with the smallest level number instead of that with the smallest rank among the ready tasks, then the total completion time may be longer. The reason for this is that the level number can be smaller than the rank number, which may delay the scheduling of key vertices. This phenomenon is shown in the following example.

Example 3.8 :

p=3 sequence	1	2	3	4	5	6	7	8	9	10	11	12
	1	4	7	10	13	16	17	20	21	22	25	27
	2	5	8	11	14		18			23	26	28
	3	6	9	12	15		19			24		

It can be seen that the completion time is 12 rather than 11 as in the previous example.

The time to finish all the tasks (the schedule length) provides a measure of processor utilization. For the strategy of scheduling based on the smallest rank number, the ratio of the schedule length and optimal schedule is bounded by 4/3 for two processors and $2 - 1/(p-1)$ for $p > 2$ processors.

There are several algorithms which give the schedule length closer to the optimal, for example, the Coffman-Graham algorithm [29]. The ratio of its schedule length and the optimal is bounded by $2 - 2/p$ where p is the number of processors. These algorithms have more complex strategies.

In practice, the analysis time for each vertex (subcircuit) cannot be the same and the number of processors available may be variable. One processor should be assigned for the sequencing task. It is assumed that this sequencing processor is efficient enough and it will not delay any task of analyzing the vertex (subcircuit). Algorithm 3.7 below gives one example of such kind of algorithm.

Algorithm 3.7

```
BEGIN
  k=1
  FOR each vertex v_i in G(V,E) DO
    BEGIN
      nu(v_i)=nfin(v_i)
      IF nu(v_i)=0
        THEN push the vertex v_i into the queue Q
    END
  REPEAT
    IF the queue Q is not empty
      THEN
        BEGIN
          the number of processors available is p
          the number of ready tasks in the queue Q is q
          IF q > p
            THEN n=p
            ELSE n=q
          k=0
          REPEAT {analyze these n vertices on n processors}
            pop out v_j from the queue Q
            k=k+1
            FOR each vertex v_i in fout(v_j) DO
              BEGIN
                nu(v_i)=nu(v_i)-1
                IF nu(v_i)=0
                  THEN push v_i into the queue Q
              END
          UNTIL k=n
        END
  UNTIL all vertices in G(V,E) are scheduled (k>m)
END
```

3.8. Discussion

In this chapter, different applications of analysis sequencing were discussed. As the task of sequencing needs to be done only once before analyzing the circuits, it will require a small portion of the total computation time. It is worthwhile to implement the sophisticated algorithms which could save computation time and increase the accuracy on the simulated results. In PREMOS, Algorithms 3.4, 3.5 and 3.6 have been implemented and the results are very satisfactory.

The scheduling algorithms for multi-processor computers depend on the characteristics and the structure of the computer and on the type of simulator being implemented. It has been shown in [23] that the speed of logic simulation could be increased more than several hundred times by using logic processors and array processors. For large-scale circuit simulation, the study of sequence scheduling on a multi-processor computer is a promising area of research.

CHAPTER 4

Nonlinear Analysis Methods

4.1. Introduction

For the circuits containing nonlinear elements, both DC analysis and transient analysis require solving sets of nonlinear algebraic equations of the form

$$g(x^n) = 0 \tag{4.1}$$

as described in Chapter 2. Eq. (4.1) is usually solved by using a modified Newton-Raphson's method. In large-scale circuit analysis, the entire circuit is partitioned into smaller subcircuits and nonlinear analysis is performed at the subcircuit level, which could provide savings in CPU time. In Section 4.2, different ways of decomposing an electronic circuit into subcircuits are discussed. DC analysis is necessary to provide the operating points at initial timepoint and is also used at every timepoint during the numerical integration procedure. Initial DC analysis in large-scale circuit simulation is described in Section 4.3. In Section 4.4 it is shown that solving the partitioned nonlinear subsystems sequentially takes much less effort than solving the entire system without partitioning. A new modified Newton's method for the DC analysis of MOS transistor circuits using a 2-element companion model for the MOS transistor (see Appendix 1) will be described in Section 4.5. The convergence

properties of the new technique are investigated in Section 4.6. The discussion and conclusion are given in Section 4.7.

4.2. Decomposition

In macromodeling approaches such as in MOTIS [6] and MOTIS-C [7], each subcircuit corresponds to a logic element such as a NAND, NOR or transfer gate. Generally this approach does not give good accuracy because interactions among logic elements by transfer gates and series drive transistor effects are not modeled sufficiently. Recently, two approaches have been proposed to decompose circuits into unilateral subcircuits. The first is to decompose the circuit into subcircuits based on a clustering algorithm applied to the circuit model in steady state; i.e. with all capacitors open-circuited. This approach works well for MOS circuits since in steady state the gate is not affected by the source or drain voltage. This approach has been used in MOS timing [8] and logic simulation [3]. For general circuits containing bipolar transistors, further approximations are needed to obtain this type of 'one-way' circuit decomposition [9]. Note that in this decomposition approach, subcircuits composed solely of transfer gates are avoided. The second approach of circuit decomposition is modular partitioning, where the circuit is composed of identifiable modules or subcircuits [30]. In the program PREMOS, a number of subcircuit types have been selected as primitive modules, which have the 'one-way' property.

4.3. Initial DC Analysis

After the procedures of decomposition and analysis sequencing are completed, the DC analysis at the initial time point is done to give appropriate initial values of voltages or currents for the transient analysis. At this initial DC analysis, all capacitors are open-circuited and their values are assumed zero. Some node voltages could have been preset as the initial guess. Instead of solving the entire circuit, the DC analysis is processed sequentially at the subcircuit level following the analysis sequence. Although this approach is a relaxation one compared to conventional circuit simulation, it could generally provide relatively accurate DC levels with reduced computational efforts. Some modification to Newton's method have been used in evaluating the DC levels at each subcircuit: (1) if the evaluated node voltage exceeds 2 Vcc, where Vcc is the power supply, then it is only recorgnized as Vcc; (2) if the computed value of node voltage is less than -Vcc, then it is set to 0. The objective is to prevent any large change of node voltage solution.

4.4. Discussion on The Convergence Rates

In this section, we want to illustrate that, by using the Newton-Raphson method, solving the partitioned nonlinear subsystems sequentially takes less effort than solving the entire system. Most of the theorems and definitions mentioned below are found in [16, 31, 32].

The spectral radius $\rho(A)$ of the n by n matrix A is defined as the maximum of the moduli of the eigenvalues of A; i.e., if $\{\lambda_i\}_{i=1,n}$ is the set of eigenvalues of A, then

$$\rho(A) = \max_{1 \leq i \leq n} | \lambda_i | \tag{4.2}$$

We consider the iterative method as a completely consistent linear stationary method of first degree, which may be expressed in the form

$$u^{(n+1)} = G\, u^{(n)} + k \qquad n = 0, 1, 2, \ldots \tag{4.3}$$

where G is the real n by n iteration matrix and k is an associated known vector.

Theorem 4.1 [32] :

The iterative method (4.3) is convergent if and only if the spectral radius $\rho(G)$ is less than one, i.e., $\rho(G) < 1$.

We define the rate of convergence, R, by

$$R(G) = - \log \rho(G) \tag{4.4}$$

Theorem 4.2 [32] :

Let the block triangular matrix T be partitioned. Then

$$\lambda(T) = \bigcup_{i=1}^{n} \lambda(T_{ii}) \tag{4.5}$$

where T_{ii} is the block diagonal matrix and $\lambda(T)$ and $\lambda(T_{ii})$ represent the set of eigenvalues of T and T_{ii} respectively.

Let the iteration matrix G be partitioned into the block lower triangular form. We could obtain the following theorem:

Theorem 4.3 :

The convergence rate of the iteration matrix G is the minimum of that of the block diagonal submatrix G_{ii}, i.e.,

$$R(G) = \min_i R(G_{ii}) \tag{4.6}$$

Proof :

It follows from Theorem 4.1 and Theorem 4.2.

For the system whose iteration matrix can be represented in the lower block triangular form, the number of iterations required for the entire matrix to converge would depend upon the maximum of that among all of its block diagonal submatrices. In practice, for many of the subvectors u_i, few iterations are needed to reach steady states. When the subsystems are solved independently in a certain sequence, the number of nonlinear iterations required to solve each subsystem depends upon its own rate of convergence. By using such an approach, the computation time could be reduced significantly. For the general systems whose iteration matrix is not in the block lower triangular form, the block Gauss-Seidel-Newton algorithm [16] could be used effectively provided that the coupling among the subsystems is not very 'strong'. Thus, to solve large-scale nonlinear system, the basic approach is, first, partition the entire system into subsystems, and then, analyze the subsystems in the proper sequence.

4.5. Numerical Properties of The Mixed Method

In conventional circuit analysis, 3-element companion models of static MOS transistors (Fig. 4.1) are generally used for representing the nonlinear operations of the device. In our approach, we assume that the gate-to-source voltage is given and thus use a 2-element companion model (Fig. 4.2) to evaluate the Jacobian matrix at each iteration. For the Gauss-Seidel algorithm with analysis sequencing, the fan-in gate voltage of pass transistor is known but the corresponding source (or drain) voltage may still be unsolved. If we use 2-element MOS transistor models by setting the unknown source voltage initially equal to its value at previous iteration the algorithm will no longer be the standard modified Gauss-Seidel method. It could be considered as a modified version of Newton-Raphson's algorithm at the subcircuit level. In the following, the results of using the 2-element model and the 3-element model for both Newton's algorithm and the standard Gauss-Seidel algorithm are compared. The convergence rate of the DC iteration is also discussed.

(i) the algorithm with the 3-element companion model

For the test circuit shown in Fig. 4.3, the nodal equations of the equivalent circuit using a 3-element model for the MOS device (Fig. 4.4) are

$$g_1(v_1-v_{cc})-i_1+g_2v_1+g_{m2}v_3+i_2+g_3(v_1-v_2)+g_{m3}(v_4-v_2)+i_3 = 0 \qquad (4.7)$$

$$g_3(v_2-v_1)-i_3-g_{m3}(v_4-v_2) = 0 \qquad (4.8)$$

The above two equations can be represented in matrix form as

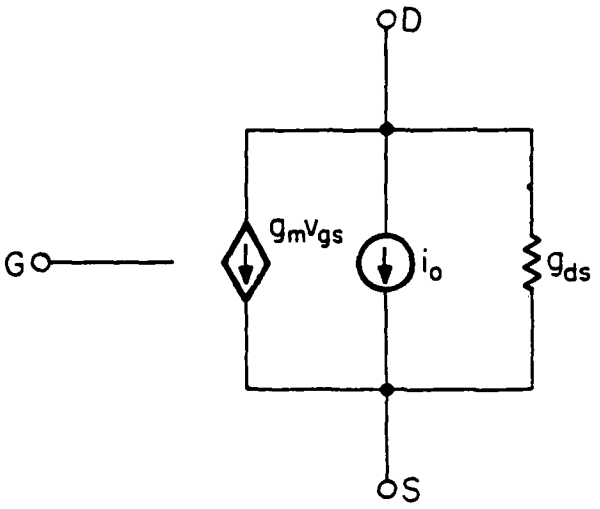


Fig. 4.1 3-Element Companion Model for MOS Transistor.

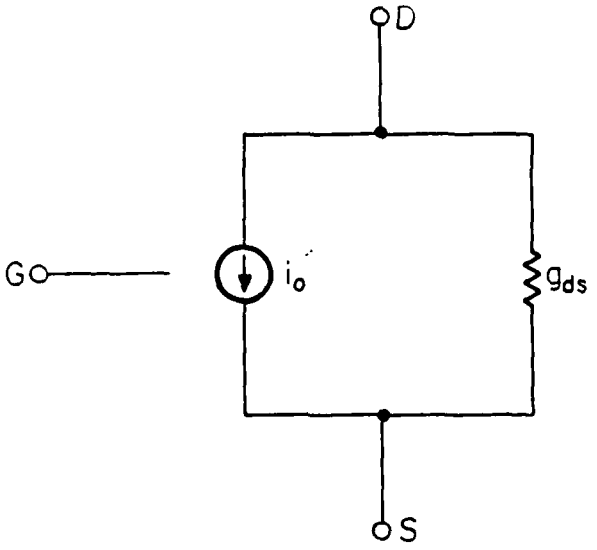


Fig. 4.2 2-Element Companion Model for MOS Transistor.

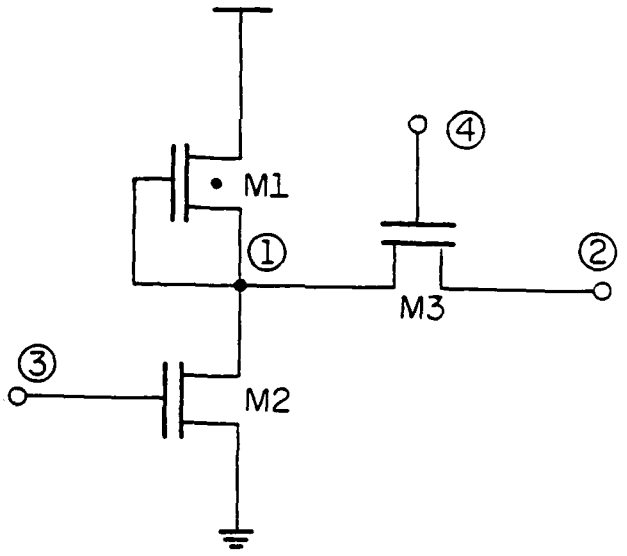


Fig. 4.3 Test Circuit.

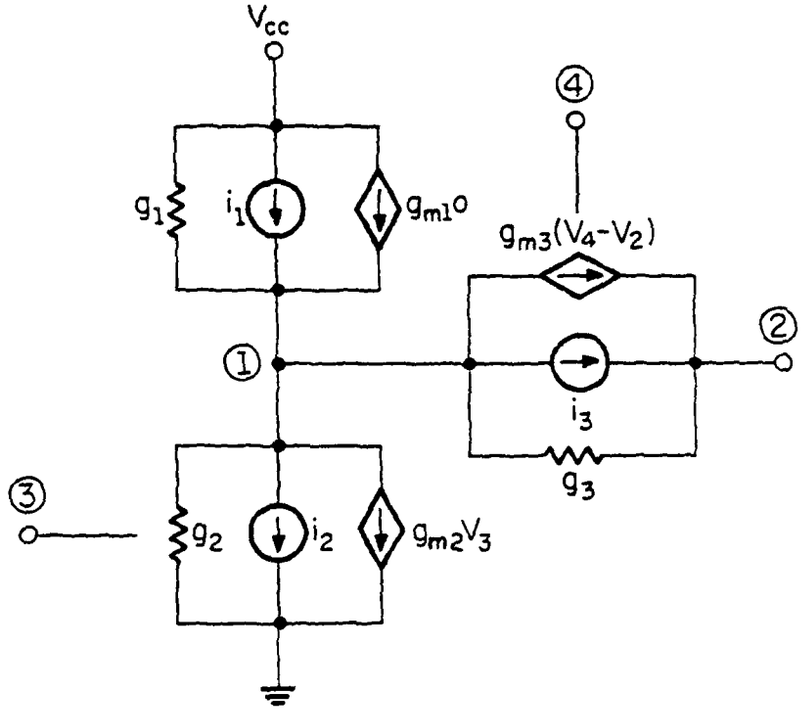


Fig. 4.4 Equivalent Circuit for Fig. 4.3 Using the 3-Element Model.

$$\begin{bmatrix} g_1+g_2+g_3 & -g_{m3}-g_3 \\ -g_3 & g_{m3}+g_3 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} g_1 v_{cc}+i_1-i_2-i_3-g_{m2}v_3-g_{m3}v_4 \\ i_3+g_{m3}v_4 \end{bmatrix} \quad (4.9)$$

or

$$M_1 \qquad V \quad = \qquad J_1$$

Solving (4.9) with the standard Gauss-Seidel method, we obtain

$$\begin{bmatrix} g_1+g_2+g_3 & 0 \\ -g_3 & g_{m3}+g_3 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} g_1 v_{cc}+i_1-i_2-i_3-g_{m2}v_3-g_{m3}v_4+(g_{m3}+g_3)v_2^p \\ i_3+g_{m3}v_4 \end{bmatrix} \quad (4.10)$$

or

$$M_1' \qquad V \quad = \qquad J_1'$$

(ii) the algorithm with the 2-element companion model

If we use the 2-element companion model for the MOS device and take the previous value of the source voltage v_2 in evaluating the model of T3, then the corresponding nodal equations of the equivalent circuit (Fig. 4.5) become

$$\begin{bmatrix} g_1+g_2+g_3 & -g_3 \\ -g_3 & g_3 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} g_1 v_{cc}+i_1'-i_2'-i_3' \\ i_3' \end{bmatrix} \quad (4.11)$$

or

$$M_2 \qquad V \quad = \qquad J_2$$

where

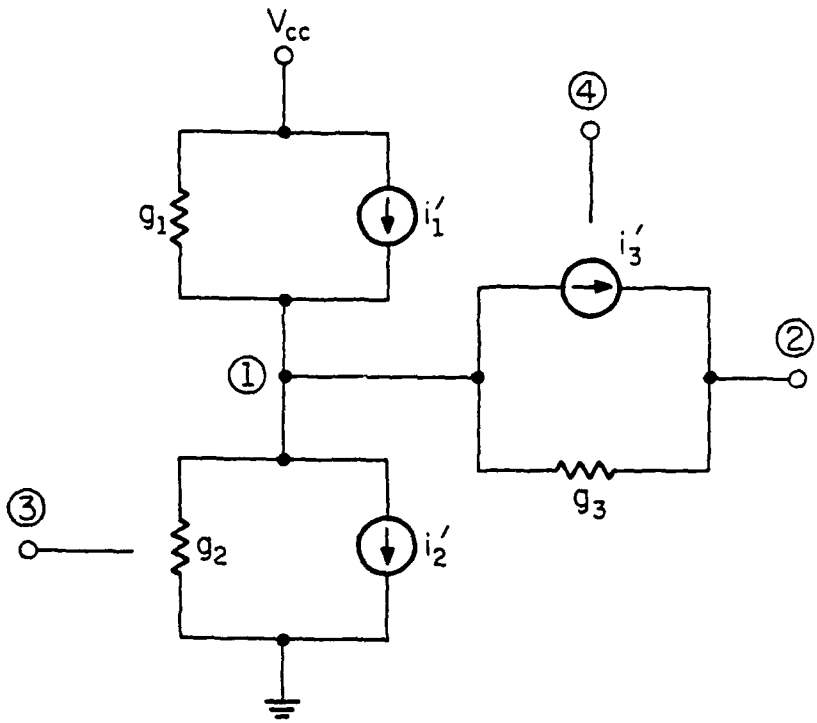


Fig. 4.5 Equivalent Circuit for Fig. 4.3 Using the 2-Element Model.

$$i_1' = i_1$$

$$i_2' = i_2 + g_{m2}v_3$$

$$i_3' = i_3 + g_{m3}v_4 - g_{m3}v_2^p$$

and v_2^p is the value of v_2 at previous iteration.

In order to solve the above equations by using the standard Gauss-Seidel method, (4.11) should be modified to

$$\begin{bmatrix} g_1+g_2+g_3 & 0 \\ -g_3 & g_3 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} g_1v_{cc}+i_1'-i_2'-i_3'+g_3v_2^p \\ i_3' \end{bmatrix} \qquad (4.12)$$

or

$$M_2' \qquad V \quad = \qquad J_2'$$

where

$$i_1' = i_1$$

$$i_2' = i_2 + g_{m2}v_3$$

$$i_3' = i_3 + g_{m3}v_4 - g_{m3}v_2^p$$

and v_2^p is the value of v_2 at previous iteration.

In the following, four methods are compared. Method 1, whose iteration matrix is M_1 *in* (4.9), *represents the conventional* Newton-Raphson method using the 3-element companion model. Method 2 with iteration formula (4.10) is the standard Gauss-Seidel method using the 3-element model. Method 3 represented by (4.11) is the modified Newton's method with the 2-element model. Method 4 with iteration formula (4.12) is the standard Gauss-Seidel method using the 2-element companion model.

Table 4.1 Rate of Convergence.

Initial Guesses (v_1, v_2, v_3, v_4) (w/l)* of T1,T2 T3	No. of DC Iterations Required			
	Method 1	Method 2	Method 3	Method 4
(1) 1.0 1.0 5.0 5.0 (4, 6, 4)	11	17	11	17
(2) 1.0 1.0 5.0 5.0 (4, 10, 1)	19	20	19	20
(3) 1.0 1.0 5.0 5.0 (4, 10, 2)	13	15	13	15
(4) 1.0 1.0 5.0 5.0 (4, 10, 4)	10	14	10	14
(5) 4.5 3.5 1.0 5.0 (4, 6, 4)	77	77	72	72
(6) 4.5 3.5 1.0 5.0 (4, 10, 1)	141	141	137	137
(7) 4.5 3.5 1.0 5.0 (4, 10, 2)	104	104	100	100
(8) 4.5 3.5 1.0 5.0 (4, 10, 4)	77	77	72	72

*In this table, (w/l) means the ratio of channel width w to channel length l of the MOS transistor.

Table 4.1 shows the rate of convergence of these four iterative methods, where the same criterion for convergence is applied. The number of iterations required for convergence depends upon the initial guess. The initial guesses are shown on the left-hand side of Table 4.1. To compare these four methods, an initial guess far from the solution is selected and a very strict criterion for convergence

is set. This gives one of the reasons why the number of DC iterations is large. The other reason is that the body effect is not taken into account in MOS device modeling, which could result in small convergence rate. According to the results shown in Table 4.1, the following points could be made :

(1) Method 3 has the minimum number of iterations required in all cases.

(2) The Gauss-Seidel method takes more or the same number of iterations compared to the Newton's method, and there is also a little discrepancy between the solutions.

(3) It was found that the smaller the w/l ratio of T3, the more the number of iterations required to reach the solution. This can be explained from the matrix structure: the smaller the g_3, the smaller the convergence rate.

4.6. The Number of Iterations Required To Achieve Convergence

In this section, a comparison is made between the analysis results obtained by using different methods for a fixed preassigned number of iterations. During transient analysis, the initial guess at any time is chosen as the previous value. Since the timestep is either small enough or controlled by the local truncation error, the initial guess is assumed to be close to the correct solution. Therefore, in practice, it is not necessary to go through as many iterations as shown in Table 4.1 to converge to the solution. Depending

upon the operating points, different numbers of iterations are required for convergence. In large-scale circuit analysis, the number of iterations could be limited to save the computation efforts in evaluating device models and in solving the matrix equations provided the analysis results are reasonably accurate. In Table 4.2, the solutions after 3 iterations using the four different methods at different operating points are shown, where 'final solution' means the solution by using Newton's method (Method 1).

From Table 4.2, the following observations can be made:

(1) Even with 3 iterations, the solutions for these four methods are good enough to satisfy the accuracy requirements of large-scale circuit simulation.

(2) From the accuracy point of view, Method 1 and Method 3 are better than Method 2 and Method 4. After considering the efforts in evaluating the device models, Method 3 seems to be the best choice.

In PREMOS, there are a number of modular subcircuits which are primitives for the entire circuit. By studying the structures of the subcircuit equations and after performing a large number of DC analyse of the individual subcircuits, a fixed number of iterations was selected for each type of subcircuit, which gave reasonably accurate results under a wide range of initial conditions. By assigning a fixed number of iterations to different subcircuits, we could eliminate checking for convergence and having to compute unnecessary additional iterations, and thus reduce the computational requirements.

Table 4.2 Analysis Results.

Initial Guesses (v_1, v_2, v_3, v_4) w/l of T1,T2 T3	(v_1, v_2) Final Solution	The Values of v_1 and v_2 After 3 Iterations			
		Method 1	Method 2	Method 3	Method 4
(1) 0.3 0.3 5.0 5.0 (4, 10, 8)	0.20624 0.20624	0.20655 0.20681	0.21352 0.21467	0.20656 0.20681	0.21352 0.21468
(2) 0.3 0.3 5.0 5.0 (4, 10, 4)	0.20624 0.20624	0.20674 0.20766	0.20979 0.21169	0.20674 0.20767	0.20979 0.21170
(3) 0.3 0.3 5.0 5.0 (4, 6, 4)	0.35008 0.35007	0.34926 0.34844	0.34545 0.34396	0.34926 0.34845	0.34545 0.34397
(4) 4.9 3.9 1.0 5.0 (4, 10, 8)	4.9966 3.9777	4.9917 3.9230	4.9912 3.9230	4.9914 3.9256	4.9914 3.9256
(5) 4.9 3.9 1.0 5.0 (4, 10, 4)	4.9966 3.9687	4.9929 3.9134	4.9927 3.9134	4.9928 3.9143	4.9928 3.9143
(6) 4.9 3.9 1.0 5.0 (4, 6, 4)	4.9979 3.9687	4.9941 3.9134	4.9939 3.9134	4.9940 3.9143	4.9940 3.9143

4.7. Discussion and Conclusion

The efforts required to evaluate the 3-element device model cost at least twice as much as that required to evaluate the 2-element device model. It has been shown that the new iteration method with

the 2-element device model gives the same or a better convergence rate compared to that of the conventional 3-element model. Furthermore, comparison of the accuracy after 3 iterations indicates that the analysis results with this new iterative method is acceptable and is still comparable to the conventional Newton's method using the 3-element model. Of course, since convergence is not being checked, it is conceivable that under certain conditions, the results could be inaccurate. More theorectical study on this nonlinear iteration method needs to be carried out.

To set a fixed number of iterations for different types of subcircuits seems to be an empirical approach. Such concepts have been implicitly used in several large-scale circuit simulators, such as using only one iteration in timing simulators [6, 7]. It is not easy to decide the optimal number of iterations required. Even for certain types of subcircuits, there are many factors involved, such as the device sizes, operating points, etc. To evaluate the number of iterations required for each subcircuit, we could have a customized formula which gives different weights to the key factors involved. For simple single-node subcircuits, like inverters or NOR gates, one iteration has been found to give fairly good results. In general, for MOS circuits consisting of two to four nodes, three iterations seem to be sufficient to solve the subcircuit.

CHAPTER 5

The Modified Gauss-Seidel Method

5.1. Introduction

In the standard Gauss-Seidel method, any feedback loop is decoupled by assuming that there is no change in the feedback loop over the integration timestep. Since previous values are used for the 'unsolved' variable, errors are introduced. Traditionally, these errors could be reduced by some relaxation techniques such as the Newton-Gauss-Seidel method [16]. However, in large-scale circuit analysis, a one-sweep approach is desired to minimize the computation time. To increase the accuracy of the analysis with only one sweep, it has been found in this research work that explicit formulas could be used to predict the 'unsolved' variables when feedback loops exist in the system. Section 5.2 introduces a modified Gauss-Seidel method using prediction. The numerical properties of the method are discussed in Section 5.3. In Section 5.4, a numerical method is used to estimate the order of convergence for the proposed modified Gauss-Seidel method. In Section 5.5, a test for checking the presence of parasitic oscillatory components in the solution is discussed. The chapter concludes with the conclusion and discussion in Section 5.6.

5.2. Modified Gauss-Seidel Method

Using the Gauss-Seidel method in analyzing a circuit, the 'feed-forward' interdependence among the subcircuits is accounted for by the analysis sequencing procedure. The 'feedback' interdependence, on the other hand, which is usually caused by feedback loops or floating capacitors or any other bilateral element connecting two subcircuits, is taken care of by using previous values. In this chapter, we introduce a new technique, the modified Gauss-Seidel method. This technique uses a forward predictor to evaluate the node voltages on the feedback loops (or the other node of a floating capacitor), rather than the previously computed values, when solving the associated subcircuits. Firstly, we consider the method using a first-order predictor with the backward Euler integration formula.

(i) feedback loops

Consider the configuration shown in Fig. 5.1, the nodal matrix for conventional circuit simulation will be

$$\begin{bmatrix} y_{11} & & & & & g_m \\ y_{21} & y_{22} & & & & \\ & y_{32} & y_{33} & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & y_{NN-1} & y_{NN} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} j_1 \\ j_2 \\ j_3 \\ \vdots \\ \vdots \\ j_N \end{bmatrix} \qquad (5.1)$$

where g_m is the transconductance at the operating point in the MOS device model.

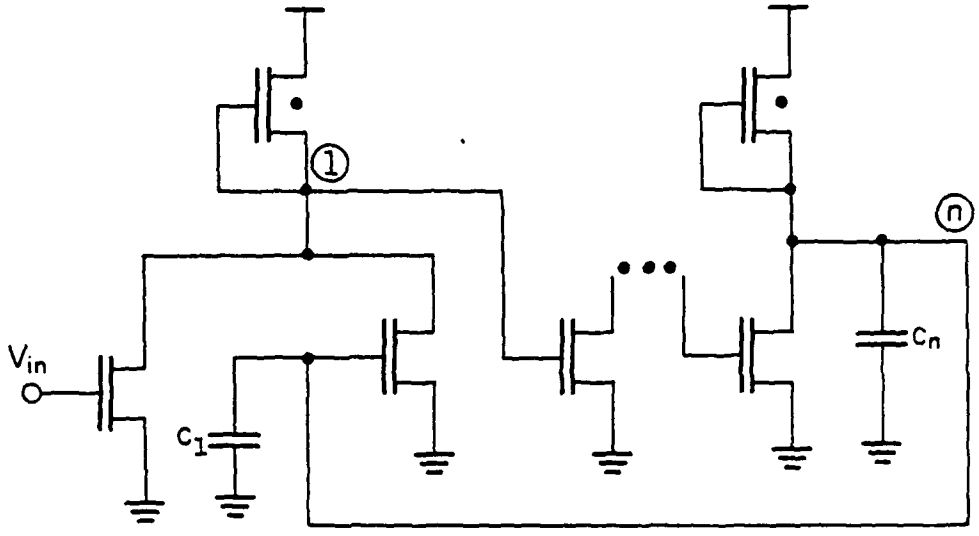


Fig. 5.1 A N-Stage Feedback Circuit.

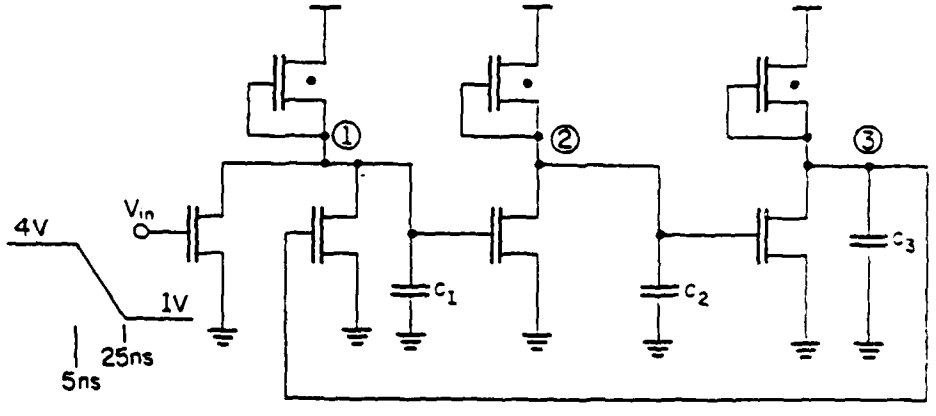


Fig. 5.2 3-Stage Ring Oscillator.

If one uses the standard Gauss-Seidel method, the g_m term in the upper triangular part caused by the feedback is moved to the right-hand side as shown in (5.2), where the term $g_m v_N$ uses the value of v_N computed at a previous iteration step.

$$\begin{bmatrix} y_{11} & & & & & \\ y_{21} & y_{22} & & & & \\ & y_{32} & y_{33} & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & y_{NN-1} & y_{NN} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} j_1' = j_1 - g_m v_N \\ j_2 \\ j_3 \\ \vdots \\ j_N \end{bmatrix} \qquad (5.2)$$

Note that the voltage v_N is also one of the controlling voltages which determines y_{11} during each iteration when solving the whole matrix. Thus, both y_{11} and j_1' are affected by the value of v_N. In the proposed approach, the value of v_N is first predicted at the present time point by using previous points $v_N^{(n)}$ and $v_N^{(n-1)}$ according to the following formula

$$v_N^{(n+1)}{}_{guess} = v_N^{(n)} + h_n \left(\frac{v_N^{(n)} - v_N^{(n-1)}}{h_{n-1}} \right) \qquad (5.3)$$

where h_n and h_{n-1} represent the present and the previous time steps, respectively.

A test program in which the above technique is implemented has been used to simulate a 3-stage ring oscillator (Fig. 5.2), which is a critical example of simulation with a feedback path. The timestep is variable and controlled by the local truncation error at each timepoint during the analysis. The results obtained are shown in

Fig. 5.3, where it can be seen that the new approach produces more accurate results than the standard Gauss-Seidel method in about the same amount of computation time.

(ii) floating capacitors

The proposed method can also be used to take into account the feedback effects of floating capacitors. For the circuit shown in Fig 5.4, the nodal equation in matrix form is

$$\begin{bmatrix} y_{11} & & -c/h \\ y_{21} & y_{22} & \\ -c/h & y_{32} & y_{33} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} j_1 \\ j_2 \\ j_3 \end{bmatrix} \tag{5.4}$$

Using the proposed method, the value of the v_3 is predicted by $v_{3\ guess}$ in solving the equation

$$y_{11}v_1 - (c/h)v_{3\ guess} = j_1 \tag{5.5}$$

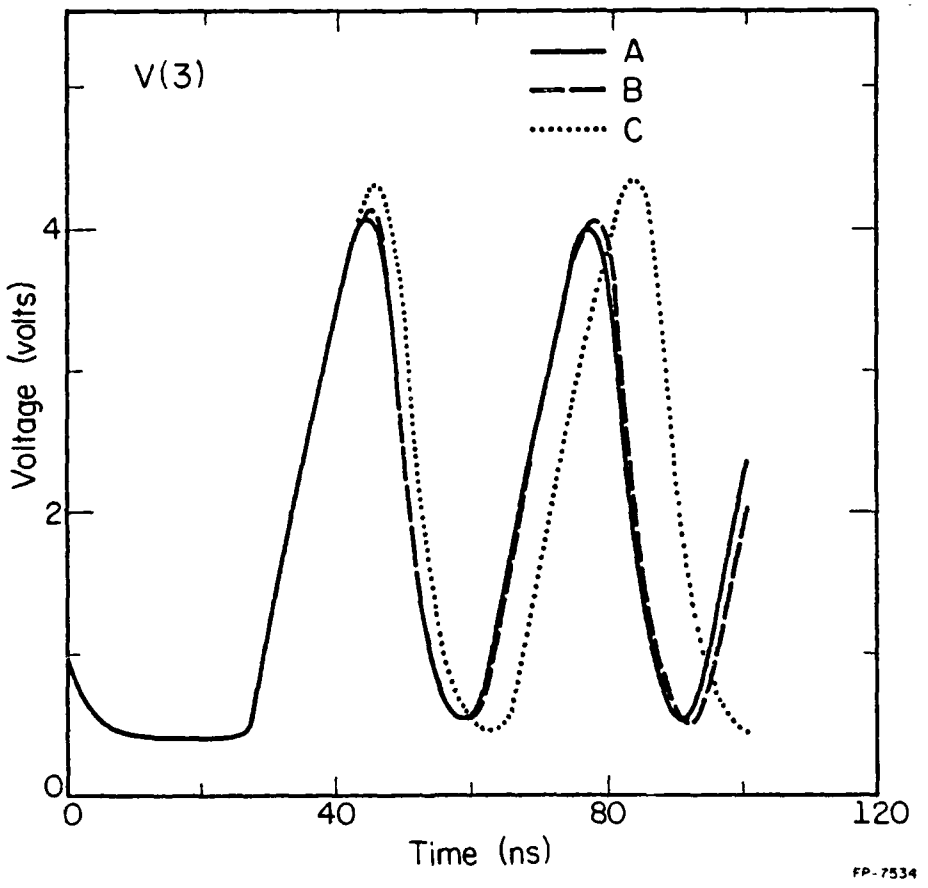
Then

$$v_1 = (y_{11})^{-1}(j_1 + (c/h)v_{3\ guess}) \tag{5.6}$$

where

$$v_{3\ guess}^{(n+1)} = v_3^{(n)} + h_n\left(\frac{v_3^{(n)} - v_3^{(n-1)}}{h_{n-1}}\right) \tag{5.7}$$

As shown in Fig. 5.[illegible] (a) and (b), this method produces more accurate results than the standard Gauss-Seidel method when compared with the circuit simulation obtained by solving the entire matrix without partitioning. In all cases, the backward Euler formula is used for numerical integration and a local truncation error timestep



A : Solving The Entire Matrix Without Partitioning

B : The Proposed Modified Gauss-Seidel Method

C : The Standard Gauss-Seidel Method

Fig. 5.3 Response of circuit of Fig. 5.2.

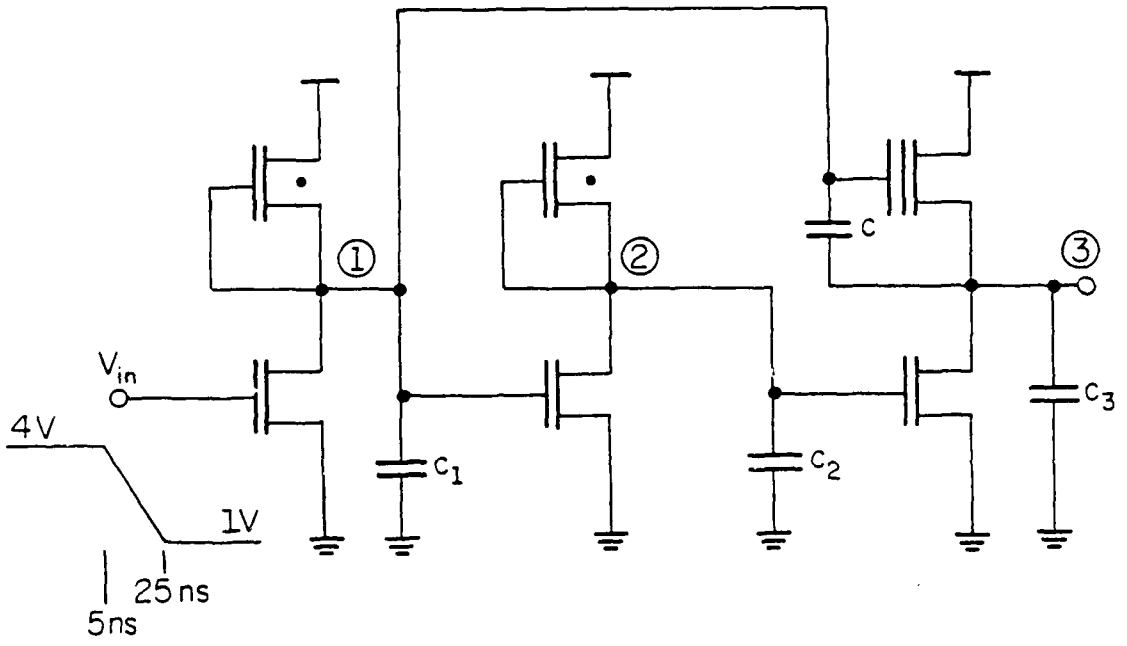
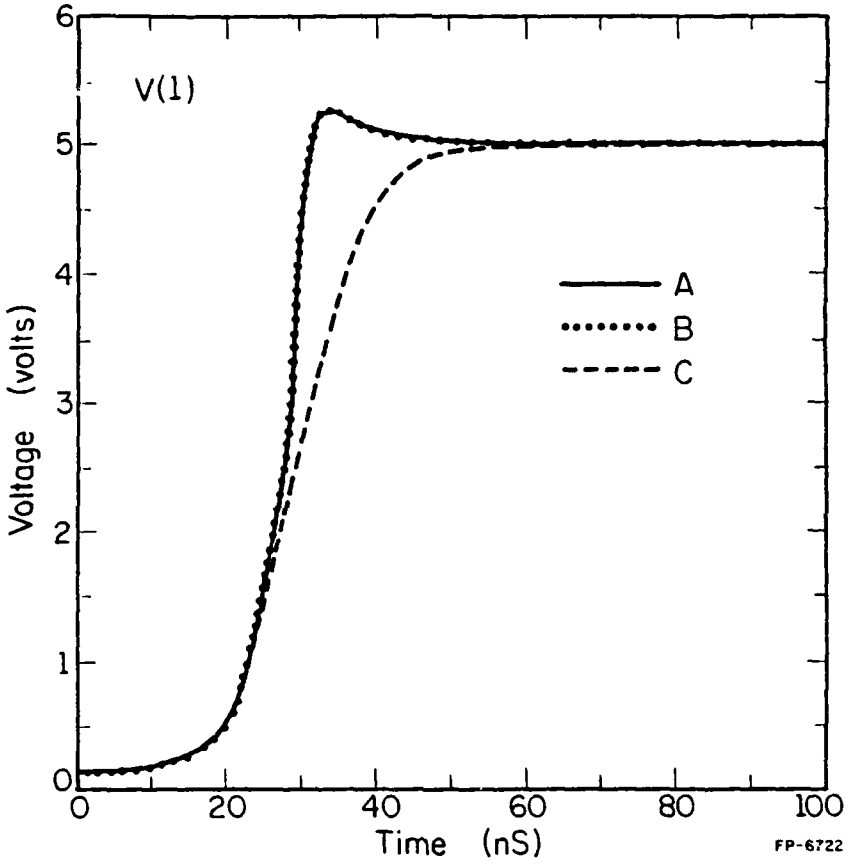


Fig. 5.4 Bootstrap Capacitor Circuit with
$c_1 = c_2 = c_3 = 13.2pf$, $c = 20.0pf$.

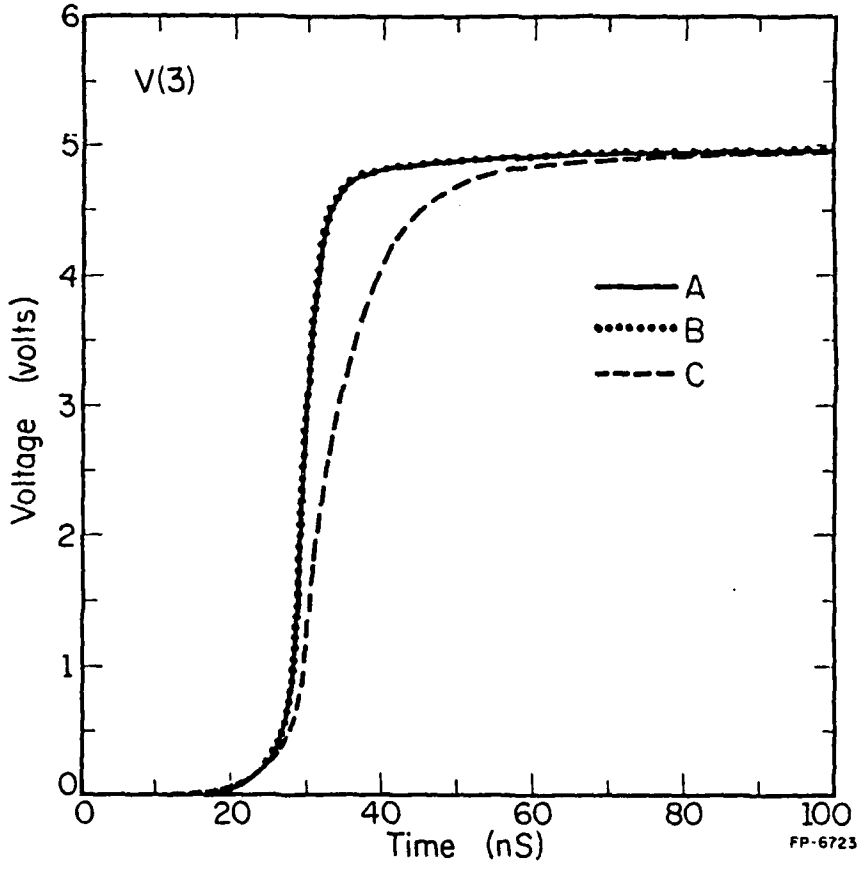


A : Solving The Entire Matrix Without Partitioning

B : The Proposed Modified Gauss-Seidel Method

C : The Standard Gauss-Seidel Method

Fig. 5.5 (a) Voltage Responses V(1) for Fig. 5.4 with the Backward Euler Integration Formula.



A : Solving The Entire Matrix Without Partitioning

B : The Proposed Modified Gauss-Seidel Method

C : The Standard Gauss-Seidel Method

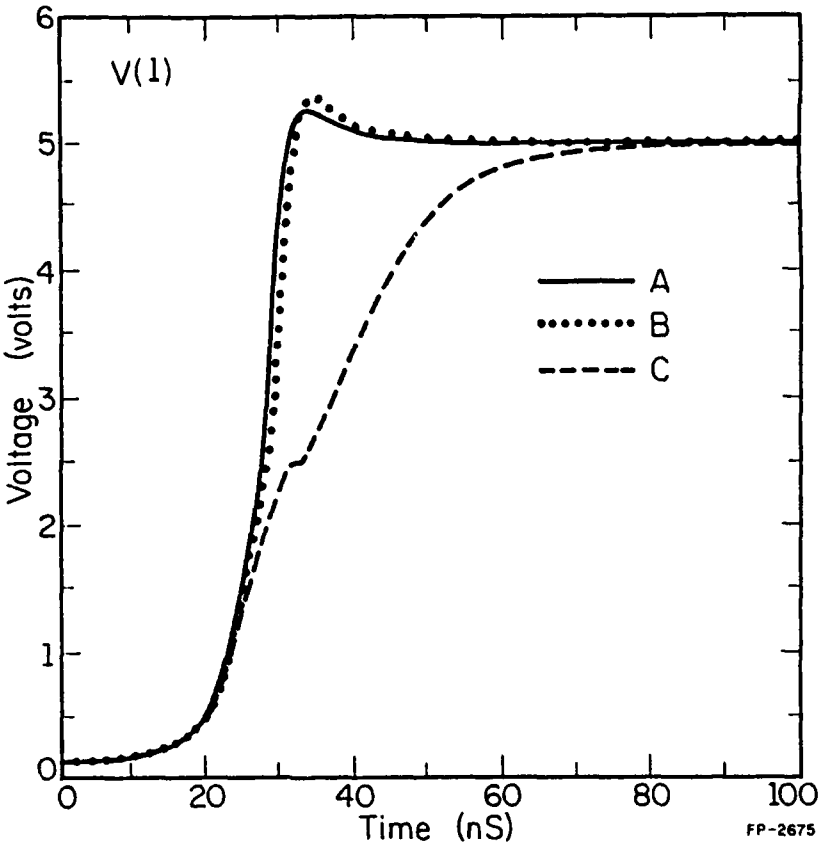
Fig. 5.5 (b) Voltage Responses V(3) for Fig. 5.4 with the Backward Euler Integration Formula.

control scheme with the absolute tolerance 1.0e-3 is employed during the integration process. With the same criterion used for checking the convergence of the nonlinear iterations and the same tolerance on the local truncation error, the number of total timepoints in this example is 427 for the circuit simulation, 398 for the standard Gauss-Seidel method and 419 for the modified Gauss-Seidel method. Since these three methods require about the same number of solution timepoints for transient analysis, they seem to have the same order of accuracy.

The modified Gauss-Seidel method has also been applied using the trapezoidal integration formula. In this case, the corresponding second-order predictor to be used for the 'unsolved' variables v_i on the feedback loops (or the other node of a floating capacitor) is of the form

$$v_{i\ guess}^{(n+1)} = v_i^{(n)} + h_n\left[(3/2)\left(\frac{v_i^{(n)} - v_i^{(n-1)}}{h_{n-1}}\right) - (1/2)\left(\frac{v_i^{(n-1)} - v_i^{(n-2)}}{h_{n-2}}\right)\right] \tag{5.8}$$

For the bootstrap capacitor circuit given in Fig. 5.4, the simulated results shown in Fig. 5.6 indicate that this second-order method is also more accurate than the standard Gauss-Seidel method. In these simulations, the timestep is determined by a local truncation error timestep control scheme in which the absolute tolerance is ε_a = 1.0e-12 and the relative tolerance ε_r = 1.0e-4 [5]. The total number of timepoints in this example is 140 for the circuit simulation, 117 for the standard Gauss-Seidel method and 127 for the modi-

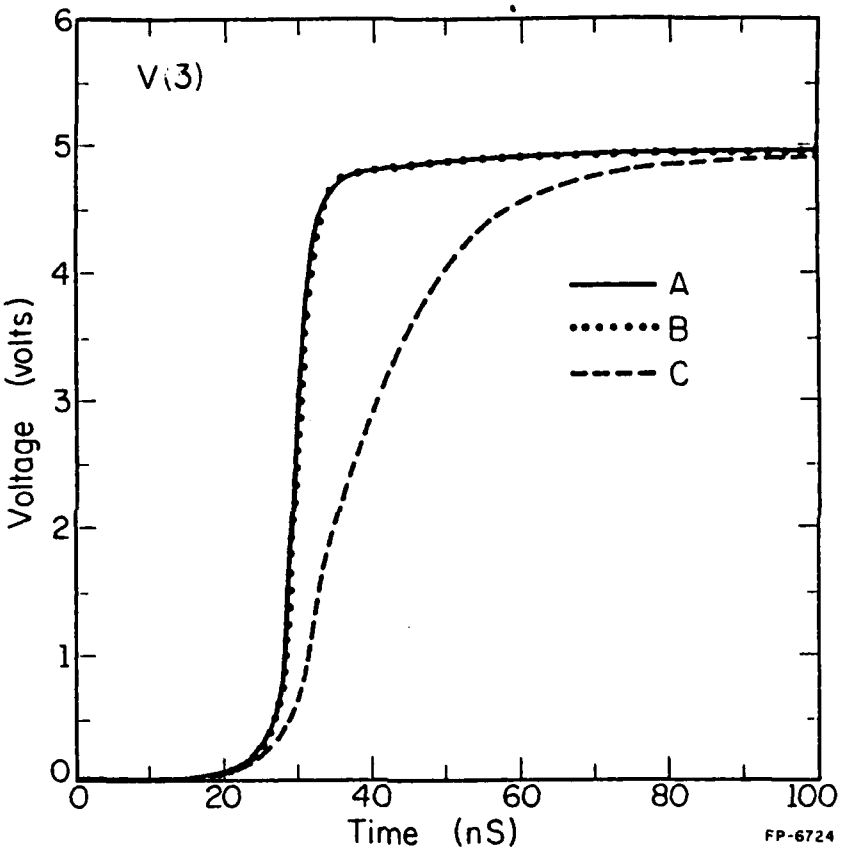


A : Solving The Entire Matrix Without Partitioning

B : The Proposed Modified Gauss-Seidel Method

C : The Standard Gauss-Seidel Method

Fig. 5.6 (a) Voltage Responses V(1) for Fig. 5.4 with the Trapezoidal Integration Formula.



A : Solving The Entire Matrix Without Partitioning

B : The Proposed Modified Gauss-Seidel Method

C : The Standard Gauss-Seidel Method

Fig. 5.6 (b) Voltage Responses V(3) for Fig. 5.4 with the Trapezoidal Integration Formula.

fied Gauss-Seidel method. Generally, with the same accuracy requirement, the analysis using the trapezoidal integration formula takes fewer timepoints than the backward Euler formula.

In summary, the modified Gauss-Seidel method is used to solve the partitioned system of equations, which now becomes a sequence of subsystem equations of the form :

$$g_k(x_1^{n+1},\cdots,x_{k-1}^{n+1},x_k,x_{k+1}^{*},\cdots,x_m^{*})=0 \tag{5.9}$$

where

$$x_i^{*}= x_i^n + h\dot{x}_i^n \tag{5.10}$$

in conjuction with the backward Euler integration formula, and

$$x_i^{*} = x_i^n + h[(3/2)\dot{x}_i^n - (1/2)\dot{x}_i^{n-1}] \tag{5.11}$$

with the trapezoidal integration formula. For a nonlinear system, eq. (5.9) is then solved using Newton's method.

5.3. Numerical Properties of The Predictor Method

To study the numerical properties of an integration method, a linear time-invariant zero-input asymptotically stable system of differential equations is chosen as the test problem, which is usually of the following form :

$$\dot{x} = Ax$$

$$x(0) = x_o \tag{5.12}$$

where $A \in R^{nxn}$ and the set of eigenvalues of A, $\sigma(A)$, is in the open left half plane. Since floating capacitors could exist in the circuit, the test problem for the modified Gauss-Seidel method should be of the form :

$$C\dot{x} = Ax$$

$$\dot{x} = C^{-1}Ax = A'x \tag{5.13}$$

where C represents the capacitance matrix and is considered to be nonsingular. The eigenvalues of $C^{-1}A$, $\sigma(C^{-1}A)$, are assumed to be in the open left plane. Let C = L+D+U where L is strictly lower triangular, D is diagonal and U is strictly upper triangular matrix. Similiarly, we have A = L' + D' + U' where L', D' and U' are defined in the same way as L, D and U.

Since either the backward Euler or the trapezoidal formula could be used to discretize the derivative operator, our discussion is separated into two parts:

(i) The modified Gauss-Seidel algorithm with the backward Euler integration formula :

The backward Euler formula is given as

$$\dot{x}_{k+1} = (1/h)(x_{k+1} - x_k) \tag{5.14}$$

where $h = t_{k+1} - t_k$ and k subscript refers to a particular time. By

applying the predictor formula

$$x_{k+1} = x_k + h\dot{x}_k \tag{5.15}$$

with the backward Euler integration formula to the test system (5.13), the following recursive relations are obtained:

$$\begin{aligned}[(D+L) - h(D'+L')]x_{k+1} &= Cx_k - Ux_{k+1} + hU'x_{k+1} \\ &= Cx_k - U[x_k + h\dot{x}_k] + hU'[x_k + h\dot{x}_k] \\ &= Cx_k - U[x_k + hC^{-1}Ax_k] + hU'[x_k + hC^{-1}Ax_k] \\ &= [(C-U) + h(U'-UC^{-1}A) + h^2U'C^{-1}A]x_k \end{aligned} \tag{5.16}$$

$$x_{k+1} = M_B(h)x_k \tag{5.17}$$

where x_k and $\dot{x}_k$ are assumed to be exact and the companion matrix

$$M_B(h) = [(C-U) - h(A-U')]^{-1}[(C-U) + h(U'-UC^{-1}A) + h^2U'C^{-1}A] \tag{5.18}$$

To study the numerical properties of the integration algorithm described in (5.17), the following definition is used [33].

Definition 5.1 :

An integration algorithm is consistent and zero-stable if its companion matrix M(h) can be expanded in power series as a function of the stepsize h as

$$M(h) = I + hA' + O(h^2) \tag{5.19}$$

Theorem 5.1 :

The modified Gauss-Seidel algorithm with the backward Euler

integration formula is consistent and zero-stable.

Proof :

The companion matrix $M_B(h)$ in (5.18) can be further expanded as

$$\begin{aligned} M_B(h) &= [I - h(C-U)^{-1}(A-U')]^{-1}[I + h(C-U)^{-1}(U'-UC^{-1}A) + h^2(C-U)^{-1}U'C^{-1}A] \\ &= [I + h(C-U)^{-1}(A-U') + O(h^2)]\ [I + h(C-U)^{-1}(U'-UC^{-1}A) + O(h^2)] \\ &= I + h(C-U)^{-1}(A - UC^{-1}A) + O(h^2) \\ &= I + hC^{-1}A + O(h^2) \\ &= I + hA' + O(h^2) \end{aligned} \tag{5.20}$$

where I is the identity matrix. Following Definition 5.1, this algorithm is consistent and zero-stable.

Theorem 5.2 :

If a linear multistep method is consistent and zero-stable, then it is convergent.

The proof of Theorem 5.2 can be found in many numerical books such as [34].

Theorem 5.3 :

The modified Gauss-Seidel algorithm with the backward Euler integration formula is convergent.

Proof :

It follows from Theorem 5.1 and Theorem 5.2.

Let x(t) be the exact solution at t. The local truncation error (LTE) is given by:

$$LTE_{k+1} = | x(t_{k+1}) - x_{k+1} | \tag{5.21}$$

If the order of the accuracy of the local truncation error is p+1, which means $LTE_{k+1} = O(h^{p+1})$, then the method is of order p.

Theorem 5.4 :

The modified Gauss-Seidel algorithm with the backward Euler integration formula is a first-order algorithm.

Proof :

The Taylor series expansion for x_k is

$$\begin{aligned} x_k &= x(t_{k+1}) - h\dot{x}(t_{k+1}) + O(h^2) \\ &= x(t_{k+1}) - hA'x(t_{k+1}) + O(h^2) \end{aligned} \tag{5.22}$$

Substituting (5.22) into our local truncation error computation, we get

$$\begin{aligned} LTE_{k+1} &= | x(t_{k+1}) - x_{k+1} | \\ &= | x(t_{k+1}) - [I + hA' + O(h^2)]x_k | \\ &= | \{I - [I + hA' + O(h^2)][I - hA' + O(h^2)]\}x(t_{k+1}) | \\ &= O(h^2) \end{aligned} \tag{5.23}$$

Therefore, it is a first order algorithm.

Although we have considered D and D' to be diagonal matrices, the above properties can be extended to the case where D and D' are

block diagonal matrices.

If we now consider the standard Gauss-Seidel algorithm with the backward Euler formula and apply it to the test system (5.13), we get :

$$[(D+L) - h(D'+L')]x_{k+1} = Cx_k - Ux_k + hU'x_k \tag{5.24}$$

$$x_{k+1} = M_{GS}(h)x_k \tag{5.25}$$

where

$$M_{GS}(h) = [(D+L) - h(D'+L')]^{-1}[(C-U) + hU']$$

Using the binary expansion, $M_{GS}(h)$ can be expressed as :

$$M_{GS}(h) = I + h(C-U)^{-1}A + O(h^2) \tag{5.26}$$

Thus, we can conclude that the standard Gauss-Seidel algorithm is not consistent when floating capacitors exist in the circuit.

For Gauss-Jacobi method, the same arguments can be made by deriving the companion matrix

$$M_{GJ}(h) = I + hD^{-1}A + O(h^2) \tag{5.27}$$

which also indicates that the Gauss-Jacobi algorithm is not consistent when floating capacitors exist in the circuit.

(ii) The modified Gauss-Seidel algorithm with trapezoidal formula :

The trapezoidal formula

$$\dot{x}_{k+1} = (2/h)(x_{k+1} - x_k) - \dot{x}_k \tag{5.28}$$

applied to the test system (5.13) yields

$$Cx_{k+1} - (h/2)Ax_{k+1} = Cx_k + (h/2)\dot{x}_k \tag{5.29}$$

The predictor for the trapezoidal integration formula has the following form:

$$x_{k+1} = x_k + h[(3/2)\dot{x}_k - (1/2)\dot{x}_{k-1}] \tag{5.30}$$

(5.30) is an explicit second-order Adams-Bashforth formula. Applying this predictor method to (5.29), we obtain

$$\begin{aligned}(D+L)x_{k+1} - (h/2)(D'+L')x_{k+1} &= Cx_k - Ux_{k+1} + (h/2)\dot{x}_k + (h/2)U'x_{k+1} \\ &= Cx_k - U[x_k + (3h/2)\dot{x}_k - (h/2)\dot{x}_{k-1}] \\ &\quad + (h/2)\dot{x}_k + (h/2)U'[x_k + (3h/2)\dot{x}_k - (h/2)\dot{x}_{k-1}]\end{aligned} \tag{5.31}$$

Assuming $\dot{x}_k$, x_{k-1} and $\dot{x}_{k-1}$ to be exact, (5.31) can be written as follows :

$$\begin{aligned}(C-U)x_{k+1} - (h/2)(A-U')x_{k+1} &= (C-U)x_k + (h/2)(A+U'-3UC^{-1}A)x_k \\ &\quad + (3h^2/4)U'C^{-1}Ax_k + (h/2)UC^{-1}Ax_{k-1} - (h^2/4)U'C^{-1}Ax_{k-1}\end{aligned} \tag{5.32}$$

and

$$\begin{aligned}x_{k+1} &= [(C-U)-(h/2)(A-U')]^{-1}[(C-U)+(h/2)(A+U'-3UC^{-1}A)+(3h^2/4)U'C^{-1}A]x_k \\ &\quad + [(C-U)-(h/2)(A-U')]^{-1}[(h/2)UC^{-1}A-(h^2/4)U'C^{-1}A]x_{k-1} \\ &= [I - (h/2)(C-U)^{-1}(A-U')]^{-1}\cdot[I + (h/2)(C-U)^{-1}(A+U'-3UC^{-1}A) + \\ &\quad (3h^2/4)(C-U)^{-1}U'C^{-1}A]x_k + [I - (h/2)(C-U)^{-1}(A-U')]^{-1}\cdot \\ &\quad [(h/2)(C-U)^{-1}UC^{-1}A - (h^2/4)(C-U)^{-1}U'C^{-1}A]x_{k-1}\end{aligned} \tag{5.33}$$

Theorem 5.5 :

The modified Gauss-Seidel algorithm with the trapezoidal integration formula is consistent and stable.

Proof :

Assuming x_k and x_{k-1} are exact, the Taylor series expansion for x_{k-1} is

$$\begin{aligned} x_{k-1} &= x_k - h\dot{x}_k + (h^2/2)\ddot{x}_k + O(h^3) \\ &= [I - hA' + (h^2/2)(A')^2 + O(h^3)]x_k \end{aligned} \tag{5.34}$$

Since the timestep h is small, we could also have the following binary expansion

$$\begin{aligned} [I-(h/2)(C-U)^{-1}(A-U')]^{-1} &= I + [(h/2)(C-U)^{-1}(A-U')] \\ &\quad + [(h^2/4)((C-U)^{-1}(A-U'))^2] + O(h^3) \end{aligned} \tag{5.35}$$

Substituting (5.34) and (5.35) into (5.33), we obtain

$$\begin{aligned} x_{k+1} &= [I + hA' + O(h^2)]x_k \\ &= M_T(h)x_k \end{aligned} \tag{5.36}$$

Following Definition 5.1, this algorithm is consistent and stable.

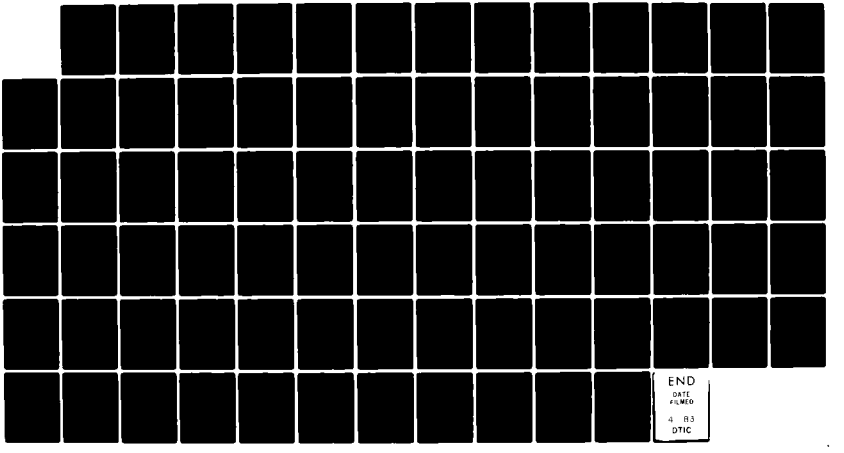
Theorem 5.6 :

The modified Gauss-Seidel algorithm with the trapezoidal integration formula is convergent.

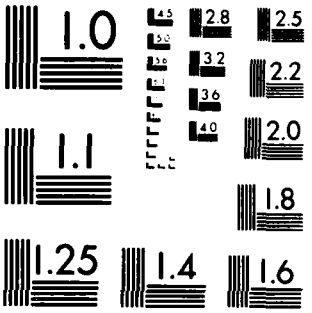
Proof :

AD-A125 941 LARGE-SCALE CIRCUIT SIMULATION(U) ILLINOIS UNIV AT URBANA COORDINATED SCIENCE LAB Y WEI DEC 82 R-977 N00014-79-C-0424 2/2

UNCLASSIFIED F/G 9/5 NL



END
DATE FILMED
4 83
DTIC



It follows from Theorem 5.5 and Theorem 5.2.

Theorem 5.7 :

The modified Gauss-Seidel algorithm with trapezoidal integration formula is of second-order.

Proof :

Assume x_k and x_{k-1} are exact and expand them into Taylor series expansion

$$x_{k+1-i} = \sum_{j=0}^{\infty} \frac{(-ih)^j}{j!} x^{(j)}(t_{k+1})$$

$$= \sum_{j=0}^{\infty} \frac{(-ih)^j}{j!} (C^{-1}A)^j x(t_{k+1}) \qquad (5.37)$$

where i=1, 2. Substituting (5.35) and (5.37) into (5.33), we obtain the local truncation error

$$LTE_{k+1} = | x(t_{k+1}) - x_{k+1} |$$

$$= O(h^3) \qquad (5.38)$$

Hence, the proposed algorithm is of second order.

5.4. Order Test

In this section, we use a numerical test to verify the order of convergence of the modified Gauss-Seidel algorithm with the backward Euler integration formula. Assume that the numerical solution x_n is related to the exact solution $x(t_n)$ through the relation

$$x_n = x(t_n) + h^p C(t_n) \qquad (5.39)$$

where p is the order of convergence. Since the dependency of $C(t_n)$ on the timestep h is of $O(h^{p+1})$, the order of convergence for the given system can be approximated by the following formula [36]

$$p = \log \left(\frac{(x_n(h) - x_n(\frac{h}{2}))}{(x_n(\frac{h}{2}) - x_n(\frac{h}{4}))} \right) / \log 2 \qquad (5.40)$$

For the linear circuit in Fig. 5.7, the order of convergence found by using the above formula is given in Table 5.1. At initial time (or time=0), V(1) = 5 V and V(2)= 0 V. Since the formula (5.40) is more valid at a smaller timestep, it can be verified that the value of p is closer to one as the timestep h gets smaller in Table 5.1. For the bootstrap capacitor circuit shown in Fig. 5.4, the results of these tests are listed in Table 5.2. Because there are nonlinear devices in this circuit, a number of iterations are required at every timepoint. The input waveform at V(1) is a step function falling from 5V to 0V in the time interval 20ns to 30ns. From Table 5.2, it is observed that the order of convergence of the modified Gauss-Seidel method is about one at 24 ns and approaches one at 28 ns with a decreasing timestep.

5.5. Accuracy Test

Applying the nodal analysis to the test circuit of Fig. 5.8, we obtain

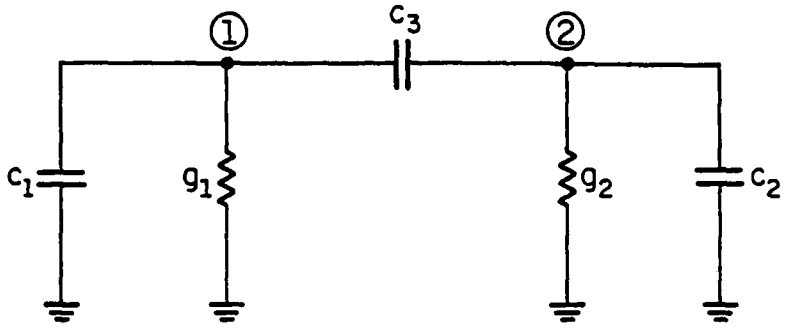


Fig. 5.7 Test Circuit ($c_1 = c_2 = c_3 = 1$, $g_1 = g_2 = 1$).

Table 5.1 Order of Convergence for the Circuit in Fig. 5.7.

Measured Point		p (h=2.0)	p (h=1.0)	p (h=0.5)
time=10	V(1)	0.74	1.00	1.01
	V(2)	0.983	0.926	0.951
time=40	V(1)	2.40	1.90	1.53
	V(2)	2.26	1.84	1.49

Table 5.2 Order of Convergence for the Circuit in Fig. 5.4.

Measured Point	Modified Gauss-Seidel Method	
V(4)	p (h=1ns)	p (h=0.5ns)
at 24 ns	1.101	1.147
at 28 ns	0.689	0.775

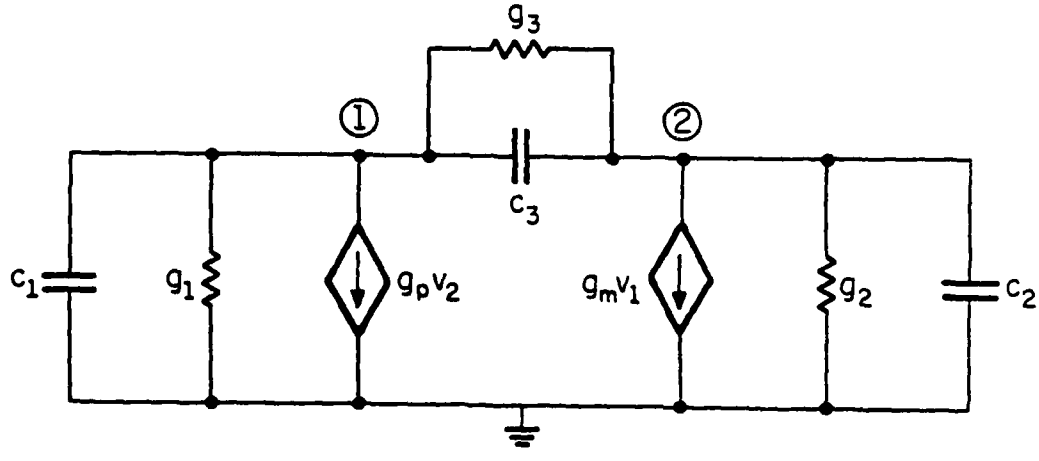


Fig. 5.8 Test Circuit.

$$c_1\dot{v}_1 + g_1v_1 + g_pv_2 + c_3(\dot{v}_1 - \dot{v}_2) + g_3(v_1 - v_2) = 0 \qquad (5.41)$$

$$c_2\dot{v}_2 + g_2v_2 + g_mv_1 + c_3(\dot{v}_2 - \dot{v}_1) + g_3(v_2 - v_1) = 0 \qquad (5.42)$$

These node equations can be represented in the following matrix form

$$\begin{bmatrix} c_1+c_3 & -c_3 \\ -c_3 & c_2+c_3 \end{bmatrix}\begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} + \begin{bmatrix} g_1+g_3 & g_p-g_3 \\ g_m-g_3 & g_2+g_3 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0 \qquad (5.43)$$

(5.43) can be rewritten in normal form

$$\begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} = A\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \qquad (5.44)$$

where

$$A(1,1)=(-(c_2+c_3)(g_1+g_3)-c_3(g_m-g_3))/DE$$

$$A(1,2)=(-(c_2+c_3)(g_p-g_3)-c_3(g_2+g_3))/DE$$

$$A(2,1)=(-(c_1+c_3)(g_m-g_3)-c_3(g_1+g_3))/DE$$

$$A(2,2)=(-(c_1+c_3)(g_2+g_3)-c_3(g_p-g_3))/DE$$

and

$$DE=c_1c_2+c_3(c_1+c_2)$$

The eigenvalues of the matrix A, λ_1 and λ_2, can be obtained by solving the characteristic equation of A. The absolute ratio of λ_1 and λ_2, which are given in Table 5.3, shows the stiffness of the test system (5.44) associated with the set of element values. By applying the backward Euler formaula

$$\dot{x}_n = \frac{x_n - x_{n-1}}{h} \tag{5.45}$$

to (5.43), we obtain

$$\begin{bmatrix} (c_1+c_3)/h + g_1 + g_3 & -c_3/h + g_p - g_3 \\ -c_3/h + g_m - g_3 & (c_2+c_3)/h + g_2 + g_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}_n +$$

$$\begin{bmatrix} -(c_1+c_3)/h & c_3/h \\ c_3/h & -(c_2+c_3)/h \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}_{n-1} = 0 \tag{5.46}$$

(i) Standard Gauss-Seidel Method

Using the standard Gauss-Seidel method in (5.46), we obtain

$$\begin{bmatrix} (c_1+c_3)/h + g_1 + g_3 & 0 \\ -c_3/h + g_m - g_3 & (c_2+c_3)/h + g_2 + g_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}_n +$$

$$\begin{bmatrix} -(c_1+c_3)/h & g_p - g_3 \\ c_3/h & -(c_2+c_3)/h \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}_{n-1} = 0 \tag{5.47}$$

Assume a solution of the form $v_{1\,n}=A_1 z^n$ and $v_{2\,n}=A_2 z^n$, where $A_1, A_2 \neq 0$. Then (5.47) is transformed into

$$\begin{bmatrix} z((c_1+c_3)/h + g_1+g_3) - (c_1+c_3)/h & g_p - g_3 \\ z(-c_3/h + g_m - g_3) + c_3/h & z((c_2+c_3)/h + g_2+g_3) - (c_2+c_3)/h \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = 0 \tag{5.48}$$

To obtain a nonzero solution the matrix in (5.48) must be singular. Thus the corresponding characteristic polynomial can be expressed as

$$Pz^2 + Qz + R = 0 \tag{5.49}$$

where

$$P=((c_1+c_3)/h+g_1+g_3)((c_2+c_3)/h+g_2+g_3)$$

$$Q=-((c_1+c_3)/h)((c_2+c_3)/h+g_2+g_3)-((c_2+c_3)/h)((c_1+c_3)+g_1+g_3)$$
$$-(-c_3/h+g_m-g_3)(g_p-g_3)$$

$$R=((c_1+c_3)/h)((c_2+c_3)/h)-(c_3/h)(g_p-g_3)$$

The sufficient and necessary condition for the nonexistence of oscillatory parasitic components in the computed solution is that the roots of (5.49) are real and positive. Critical timestep h_{crit} is defined as the maximum timestep at and below which all roots of the associated characteristic polynomial are real and positive. If $Q^2-4PR \geq 0$, then the roots are real; otherwise, they become complex. Whether the roots are positive will depend on the circuit parameters; for example, when $g_p - g_3 = 0$ in the given test circuit, the roots are always real and positive and thus there is no limit on h_{crit}. In general, a table look-up method can be used to find h_{crit} if it exists. For each set of element values shown in Table 5.3, the correspondent h_{crit} is infinite in the cases #1 to #10 and at least larger than 1.0e+4 for #11 and #12.

(ii) Modified Gauss-Seidel Method

By applying the modified Gauss-Seidel method in (5.46), we obtain

$$\begin{bmatrix} (c_1+c_3)/h + g_1+g_3 & 0 \\ -c_3/h + g_m-g_3 & (c_2+c_3)/h + g_2+g_3 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}_n +$$

$$\begin{bmatrix} -(c_1+c_3)/h & -c_3/h + 2g_p-2g_3 \\ c_3/h & -(c_2+c_3)/h \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}_{n-1} +$$

$$\begin{bmatrix} 0 & c_3/h - g_p+g_3 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}_{n-2} = 0 \qquad (5.50)$$

Assume a solution of the form $v_{1\,n} = A_1 z^n$ and $v_{2\,n} = A_2 z^n$, where A_1, $A_2 \neq 0$. Then

$$\begin{bmatrix} z^2((c_1+c_3)/h + g_1+g_3) - z((c_1+c_3)/h) & z(-c_3/h + 2g_p-2g_3) + c_3/h - g_p+g_3 \\ z(-c_3/h + g_m-g_3) + c_3/h & z((c_2+c_3)/h + g_2+g_3) - ((c_2+c_3)/h) \end{bmatrix}\begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = 0 \qquad (5.51)$$

Again, this matrix must be singular. Thus, the following characteristic polynomial can be obtained :

$$Pz^3 + Qz^2 + Rz + S = 0 \qquad (5.52)$$

where

$$P=((c_1+c_3)/h + g_1+g_3)((c_2+c_3)/h + g_2+g_3)$$

$$Q=-((c_2+c_3)/h)((c_1+c_3)/h + g_1+g_3) - ((c_1+c_3)/h)((c_2+c_3)/h + g_2+g_3)$$
$$-(-c_3/h + 2g_p-2g_3)(-c_3/h + g_m-g_3)$$

$$R=((c_1+c_3)/h)((c_2+c_3)/h) - (c_3/h)(-c_3/h + 2g_p-2g_3)$$
$$-(-c_3/h + g_m-g_3)(c_3/h - g_p+g_3)$$

$$S=-(c_3/h)(c_3/h - g_p+g_3)$$

The sufficient and necessary condition for the nonexistence of oscillatory parasitic components in the computed solution is that the

roots of (5.52) are real and positive. Given the set of element values, if the critical timestep h_{crit} exists then at h_{crit} we have

$$q^3 + r^2 = 0 \tag{5.53}$$

where

$$q = \frac{R}{3P} - \frac{1}{9}\left(\frac{Q}{P}\right)^3$$

$$r = \frac{1}{6}\left(\frac{QR}{P^2} - 3\frac{S}{P}\right) - \frac{1}{27}\left(\frac{Q}{P}\right)^3$$

If $q^3+r^2 \leq 0$ then all the roots are real, otherwise a pair of conjugate roots exists [37]. Whether the the roots are positive will depend on the circuit parameters. Therefore, a table look-up method can be used to find the h_{crit} by observing the transistion of the q^3+r^2 *from negative to positive with increasing timestep* h.

In Table 5.3, the h_{crit} is listed at different element values for the modified Gauss-Seidel method. λ_1 and λ_2 are two eigenvalues of the test system (5.44). The following conclusions can be made from this table:

(1) The effect of increasing the floating capacitor c_3 is to decrease the h_{crit}, and vice versa.

(2) Increasing the conductance g_1 slightly lowers h_{crit}.

(3) Decreasing the conductance g_2 slightly increases h_{crit}.

(4) The effects of g_m on the h_{crit} is more critical than the other parameters in the circuit. The larger g_m is, the smaller h_{crit} is.

(5) The existence of the transconductance of the feedback current, g_p, has a positive effect of increasing h_{crit}.

Table 5.3 h_{crit} for the Modified Gauss-Seidel Method.

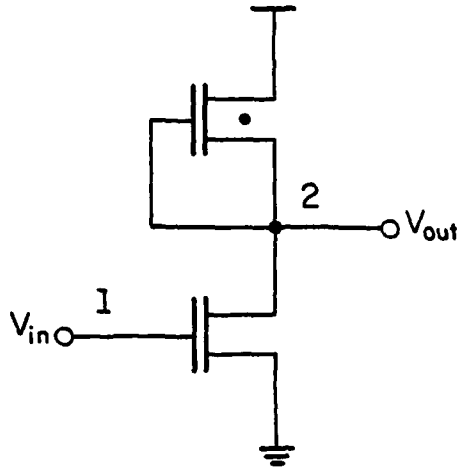
#	c_1	c_2	c_3	g_1	g_2	g_3	g_m	g_p	$\lvert\lambda_1/\lambda_2\rvert$	h_{crit}
0	1	1	0.1	0	0.1	0	1	0	0	9.2
1	1	1	1	0	0.1	0	1	0	0	0.91
2	1	1	0.01	0	0.1	0	1	0	0	95
3	1	1	0.1	1	0.1	0	1	0	0.0818	6.3
4	1	1	0.1	0.1	0.1	0	1	0	0.1568	8.6
5	1	1	0.1	0	1	0	1	0	0	6.4
6	1	1	0.1	0	0.01	0	1	0	0	9.9
7	1	1	0.1	0	0	0	1	0	0	10
8	1	1	0.1	0	0.1	0	10	0	0	0.99
9	1	1	0.1	0	0.1	0	0.1	0	0	64
10	1	1	0.1	0	0.1	0	0	0	0	320
11	1	1	0.1	0	0.1	0	1	1	0.7542	**
12	1	1	0.1	0	0.1	0	1	0.1	0.5353	**

** means h_{crit} at least larger than 1.0e+4

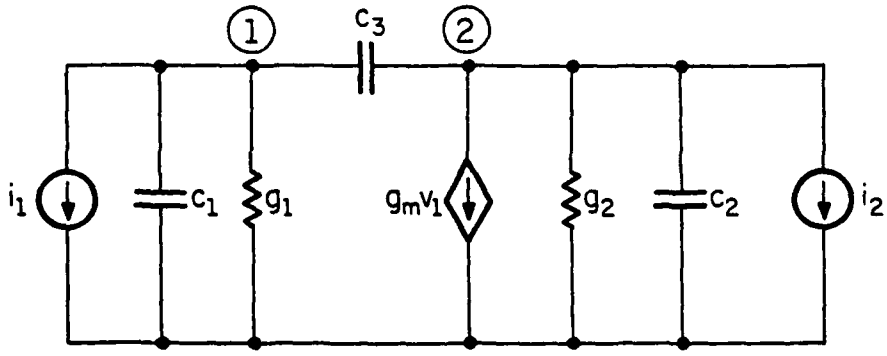
For MOS inverter circuit, whose companion circuit model is described in Fig. 5.9 but with $g_p = g_3 = 0$, the small signal gain can be expressed as

$$A_v = v_2/v_1 = g_m/g_2 \quad .$$

The typical value for g_m is 0.7e-3, and that for g_2 is from 0 to 2.0e-3. From the above table, we could figure out the range of the h_{crit} for simulating the inverter circuit.



(a)



(b)

Fig. 5.9 (a) An MOS Inverter Gate.
(b) Companion Circuit Model for (a).

5.6. Discussion

The modified Gauss-Seidel method discussed in this chapter is used to solve the circuit equations and to decouple the feedback terms during the analysis. The technique of using a forward predictor to estimate the values of the yet unsolved variables in the feedback loops was found to be more accurate than the standard Gauss-Seidel method, without requiring much additional computation. Provided the timestep is less than a critical maximum timestep h_{crit} associated with the set of element values, the accuracy test proves that no oscillatory parasitic components are presented in the computed solution. In general, for a wide range of element values, the associated h_{crit} is relatively large. So this algorithm is reasonably accurate.

CHAPTER 6

Latency and Time-Step Control Scheme

6.1. Latency Scheme

6.1.1. Introduction

During the analysis of large-scale partitioned networks, a large portion of the subnetworks is not active at any given time. This temporary inactive behavior of a subnetwork is defined as 'latency' [13, 38]. The latent status of a subsystem can be established by monitoring the changes of all its stimuli and all its responses to ensure their being within certain predetermined errors. Once the latent status of a subsystem is established, the analysis of that latent subsystem can be bypassed, and thus provide savings in CPU time.

In [14], the latency at the subnetwork level was exploited and four schemes of determining the latency in time were proposed. For the one-way macromodelling approach, all subcircuits are unilateral and each subcircuit can be identified as an 'event'. Therefore, we can take maximum advantage of the latency in time to achieve computational effiency.

6.1.2. Latency Scheme for The Network Composed of Unilateral Subnetworks

For the subnetwork N_k, let the fanin node voltages be denoted by v_{ik_p}, p=1,2,... , the internal node voltages of N_k be denoted by v_{ok_q}, q=1,2,... .

Latency Scheme :

A subnetwork N_k is considered to be latent at time t_n if

$$(1) \quad | v_{ik_p}(t_n) - v_{ik_p}(t_{n-1}) | < \varepsilon_1$$
$$(2) \quad | v_{ok_q}(t_n) - v_{ok_q}(t_{n-1}) | < \varepsilon_2$$
$$p = 1, 2,... \qquad q = 1, 2,... \tag{6.1}$$

The subnetwork N_k will remain latent at time t_{n+1} as long as

$$| v_{ik_p}(t_{n+1}) - v_{ik_p}(t_n) | < \varepsilon_1$$
$$p = 1, 2,... \tag{6.2}$$

6.1.3. Examples

The latency scheme described in Section 6.1.2 has been successfully implemented into the program PREMOS. The results of applying this latency scheme to the transient analysis of 2-bit adder, binary-to-octal decoder and 10-stage inverter chain are given in Table 6.1. The data in Table 6.1 corresponds to the error tolerance ε_1=1.0e-2 and ε_2=1.0e-3.

Table 6.1 Simulation Data for Transient Analysis.

Circuits	With Latency	Without Latency	Percentage Savings
2-Bit Full Adder (Fig. 6.1)	18.050 sec	25.867 sec	30.2 %
Binary-to-Octal Decoder (Fig. 6.2)	11.417 sec	17.583 sec	35 %
10-Stage Inverter Chain (Fig. 6.3)	3.850 sec	6.417 sec	40 %

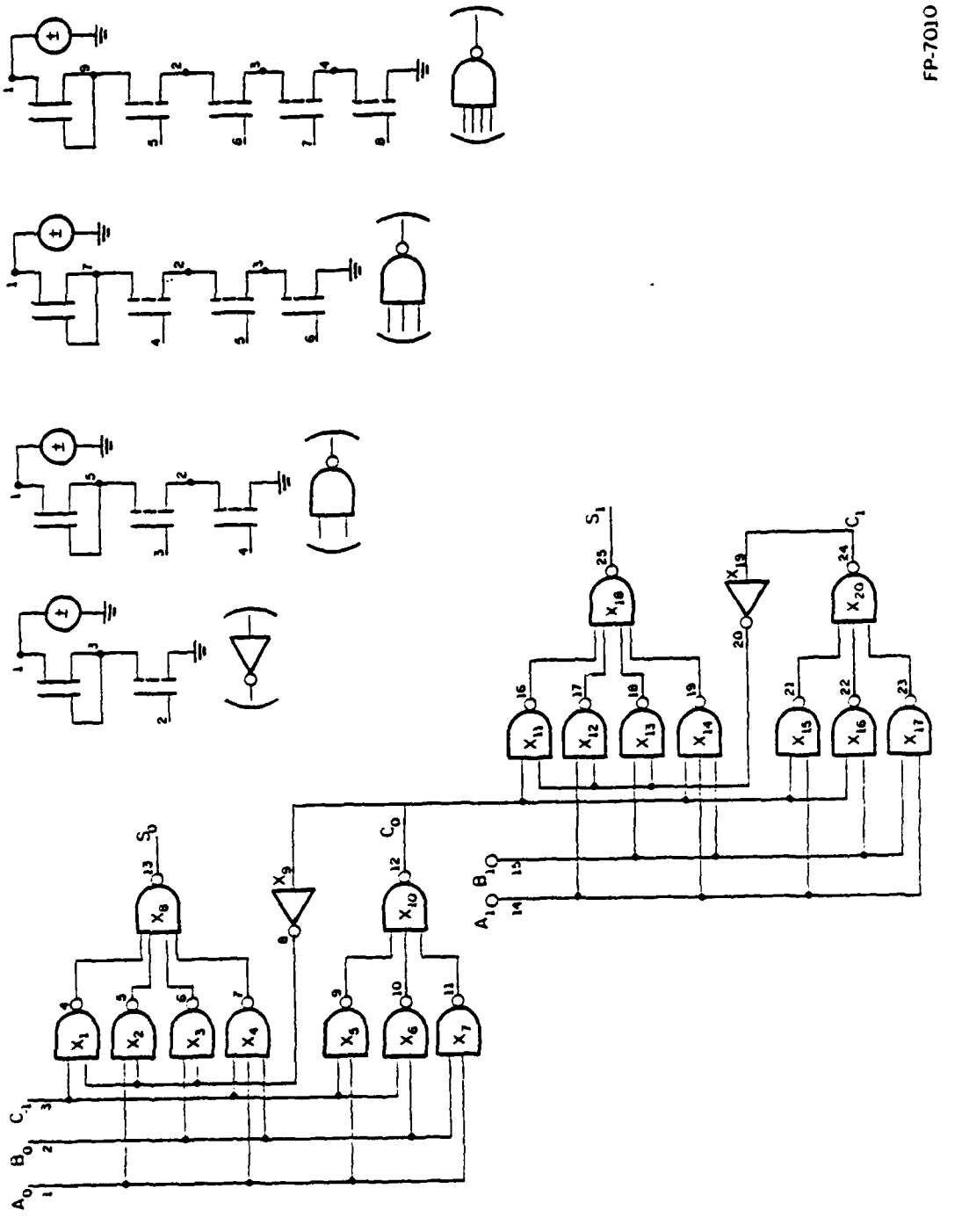


Fig. 6.1 2-Bit Full Adder.

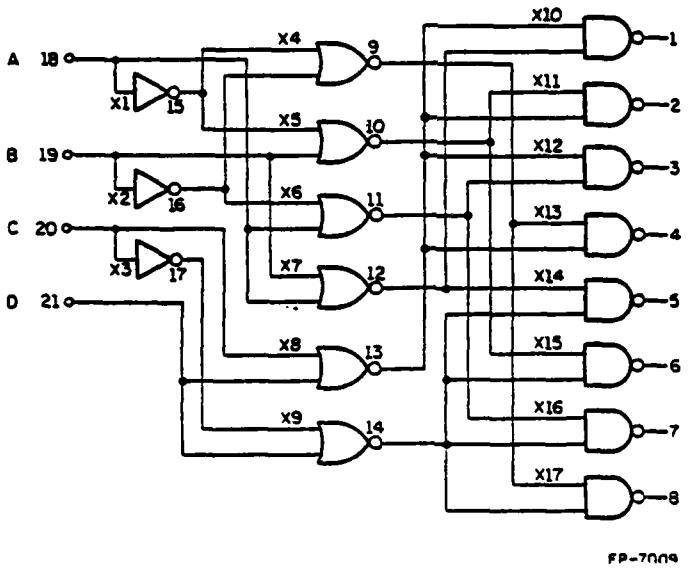


Fig. 6.2 Binary-to-Octal Decoder.

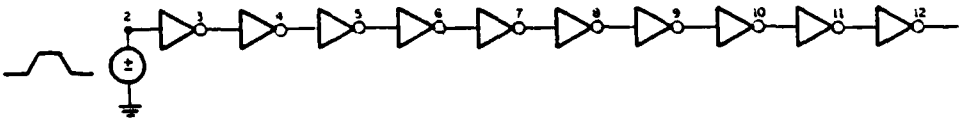


Fig. 6.3 10-Stage Inverter Chain.

6.1.4. Discussion

The latency described above is at the subnetwork level. This latency principle could be applied to any cluster of subnetworks in a large network. In [13], a multilevel latent path algorithm is presented and latency is exploited with modularities. For the programs having a multilevel macromodel structure, this approach could achieve more computational savings.

Analysis sequencing for the relevant parts has been discussed in Chapter 3. During the simulation, only the relevant parts of the circuit need to be analyzed even when the remaining parts are active. In PREMOS, this approach combined with the latency technique is employed to provide more savings in computation and in memory.

6.2. Time-Step Control Scheme

6.2.1. Introduction

In circuit simulation, the stepsize is in general determined by the local truncation error. The local truncation error of a numerical integration algorithm for solving the initial-value problem

$$\dot{x} = f(x,t),\ x(0) = x_o \tag{6.3}$$

is defined as

$$\varepsilon_T(t_{n+1}) = |x(t_{n+1}) - x_{n+1}| \tag{6.4}$$

where $x(t_{n+1})$ is the exact solution $x(t)$ to Eq. (6.3) evaluated at

$t=t_{n+1}$, and x_{n+1} is the corresponding numerical solution obtained at the same time $t=t_{n+1}=t_n + h$, provided that in using the numerical integration algorithm we assume that $x_n=x(t_n)$ is the exact solution at $t=t_n$. In other words, the local truncation error is the error made in one timestep.

The tolerance on the local truncation error is defined as

$$UT = h_n \, ED \tag{6.5}$$

where ED is the absolute value of the error allowed per unit time. In (6.5), ED is an absolute tolerance. In practice, however, relative tolerances are more meaningful. A larger ED is allocated for the fast transient part and a smaller ED for the slower transient part. After adding a relative tolerance, UT in (6.5) becomes

$$UT = h_n(\varepsilon_r|\dot{x}_{n+1}| + \varepsilon_a) \tag{6.6}$$

In (6.6), ε_r is the relative tolerance and ε_a is the absolute tolerance.

For the backward Euler integration formula, the local truncation error is evaluated by

$$\varepsilon_T(t_{n+1}) = -\frac{h_n^2}{2}\ddot{x}(\xi) \tag{6.7}$$

where $t_n \leq \xi \leq t_{n+1}$. The second derivative $\ddot{x}(\xi)$ in (6.7) can be approximated by using the divided difference formula

$$\ddot{x}(\xi) = 2 \, DD2 \tag{6.8}$$

and

$$DD2 = \frac{\frac{x_{n+1} - x_n}{h_n} - \frac{x_n - x_{n-1}}{h_{n-1}}}{h_n + h_{n-1}} \qquad (6.9)$$

If the local truncation error for timepoint t_{n+1} is considered satisfactory (less than the allowable tolerance), the new timestep h_{n+1} to compute the timepoint t_{n+2} is increased. This feature allows the solution to be found within the specified accuracy in fewer timesteps. On the other hand, if the local truncation error is too large, then the timepoint t_{n+1} is recomputed by using the reduced step-size. Therefore, the LTE at each timepoint is maintained within the specified bounds.

6.2.2. Relaxed Version of Time-Step Control Scheme

For the timestep control scheme described in the last section, if the old timestep h_n is rejected, the solution at t_{n+1} needs to be evaluated again at the newly reduced timestep h_n. Generally, this re-evaluation adds to the overhead needed for the timestep control scheme and increases total computation time. Also, due to the inaccuracy of the divided difference approximation to $\ddot{x}(\xi)$, some undesirable situations can occur if one does not implement the timestep control properly [5, 14].

In large-scale circuit simulation, most of the current programs either use a fixed timestep or a variable timestep controlled by the circuit activity (the maximum voltage variation) [18]. To have an accurate and efficient timing analysis, it may be worthwhile to have the timestep controlled by the local truncation error. However, some

modifications need to be done in order to reduce the overhead required in the implementation. The desired features for a new timestep control scheme are described in the following:

(1) The next timestep h_{n+1} is controlled by the local truncation error evaluated at the present timepoint t_{n+1}. This timestep is accepted all the time, even when it is judged to be too large.

(2) The increased timestep is double the present one and the reduced timestep is half. As shown later, this scheme ensures that the local truncation error is generally within a reasonable range of the specified tolerance even though there is no rejection of timesteps.

The timestep control algorithm is as follows :

```
BEGIN
  BEGIN
    x_n = node voltage at present timepoint t_n
    x_n-1 = node voltage at previous timepoint t_n-1
    h_n-1 = t_n - t_n-1
    h_n-2 = t_n-1 - t_n-2
    a = h_n/h_n-1 where h_n is the next timestep
    facmax=0.0
  END {inilization}
  FOR each node voltage x_n DO
    BEGIN

      ed = epa + epr * |ẋ_n|

      factor = |(ẋ_n - ẋ_n-1)|/ed
      facmax = max(facmax, factor)
    END {finding the maximum of LTE/UT}
  BEGIN
    emax = facmax * h_n-1 / (h_n-1 + h_n-2)
    IF (emax > 1.2)
      THEN a = 0.5
      ELSE
        BEGIN
          IF (emax < 0.4)
            THEN a = 2.0
            ELSE a = 1.0
```

END

END

$h_n = a * h_{n-1}$

END

It should be noted that in the above algorithm the upper limit of the local truncation error for reducing the timestep is 1.2 h_n ED, and the lower limit for increasing the timestep is 0.4 h_n ED. Although there is no rejection of the timestep at any timepoint, the test equation below shows that the local truncation error stays within the desired tolerance.

Let us consider the test equation

$$\dot{x} = \lambda x \tag{6.10}$$

where λ is negative. The exact solution of (6.10) is

$$x = x_o e^{\lambda t} \tag{6.11}$$

where x_o is the solution at time zero. Applying the backward Euler integration formula to (6.11), we obtain

$$x_{n+1} = x_n \left[\frac{1}{1 - h_n \lambda} \right] \tag{6.12}$$

Consider the situation $h_{n-2} = h_{n-1} = h$ and $h_n = a\,h$, where a could be 2, 1, or 0.5. From Eqs. (6.12), we obtain

$$\frac{LTE_{n+1}}{LTE_n} = \frac{DD2_{n+1}}{DD2_n} \frac{h_n^2}{h_{n-1}^2} = \frac{2a}{a+1} * a^2 * \frac{1}{1 + \frac{ah}{-\lambda}} \tag{6.13}$$

There are three cases to consider, depending on the value of a :

(1) a = 2

From (6.13), LTE_{n+1} is evaluated as

$$LTE_{n+1} = \frac{16}{3} \frac{1}{1+(2h/(-\lambda))} LTE_n \qquad (6.14)$$

Since $LTE_n \leq 0.4\ h_{n-1}$ ED , then

$$LTE_{n+1} \leq \frac{3.2}{3} \frac{1}{1+(2h/(-\lambda))} h_n \text{ ED} < 1.2\ h_n \text{ ED} \qquad (6.15)$$

(2) a = 1

At the previous timepoint t_n, $LTE_n \leq 1.2\ h_{n-1}$ ED. The following relation can be easily derived :

$$LTE_{n+1} \leq \frac{1}{1+(h/(-\lambda))} h_n \text{ ED} < 1.2\ h_n \text{ ED} \qquad (6.16)$$

(3) a = 0.5

In this case, LTE_n is just greater than $1.2\ h_{n-1}$ ED and $LTE_{n-1} \leq 1.2\ h_{n-2}$ ED, where $h_{n-1}=h_{n-2}$. Let $LTE_n = m * (1.2\ h_{n-1}$ ED). We assume that the second derivative $\ddot{x}(\xi)$ is continuous and varies slowly for most of the circuits. This is generally true, especially for digital circuits. Therefore, the value of m is assumed to be less than 3.6. Then, we obtain

$$LTE_{n+1} = \frac{m}{3} \frac{1}{1+(0.5h/(-\lambda))} h_n \text{ ED} < 1.2\ h_n \text{ ED} \qquad (6.17)$$

In summary, this new algorithm gives the required accuracy for the given test equation.

With $\lambda=-10$, $x_o=1$ and $h=0.01$, Table 6.2 shows the results of Backward Euler integration with a fixed timestep. For the new

Table 6.2 Numerical Solution Obtained by Using Fixed Timestep with $\lambda=-10$, $x_0=1$ and $h=0.01$.

	Time	Exact Sol.	Bk Euler	Glb Err.	LTE
0	.0000e+00	.1000e+01	.1000e+01	.0000e+00	.0000e+00
1	.1000e-01	.9048e+00	.9091e+00	-.4253e-02	-.4253e-02
2	.2000e-01	.8187e+00	.8264e+00	-.7716e-02	-.3867e-02
3	.3000e-01	.7408e+00	.7513e+00	-.1050e-01	-.3515e-02
4	.4000e-01	.6703e+00	.6830e+00	-.1269e-01	-.3196e-02
5	.5000e-01	.6065e+00	.6209e+00	-.1439e-01	-.2905e-02
6	.6000e-01	.5488e+00	.5645e+00	-.1566e-01	-.2641e-02
7	.7000e-01	.4966e+00	.5132e+00	-.1657e-01	-.2401e-02
8	.8000e-01	.4493e+00	.4665e+00	-.1718e-01	-.2183e-02
9	.9000e-01	.4066e+00	.4241e+00	-.1753e-01	-.1984e-02
10	.1000e+00	.3679e+00	.3855e+00	-.1766e-01	-.1804e-02
11	.1100e+00	.3329e+00	.3505e+00	-.1762e-01	-.1640e-02
12	.1200e+00	.3012e+00	.3186e+00	-.1744e-01	-.1491e-02
13	.1300e+00	.2725e+00	.2897e+00	-.1713e-01	-.1355e-02
14	.1400e+00	.2466e+00	.2633e+00	-.1673e-01	-.1232e-02
15	.1500e+00	.2231e+00	.2394e+00	-.1626e-01	-.1120e-02
16	.1600e+00	.2019e+00	.2176e+00	-.1573e-01	-.1018e-02
17	.1700e+00	.1827e+00	.1978e+00	-.1516e-01	-.9257e-03
18	.1800e+00	.1653e+00	.1799e+00	-.1456e-01	-.8415e-03
19	.1900e+00	.1496e+00	.1635e+00	-.1394e-01	-.7650e-03
20	.2000e+00	.1353e+00	.1486e+00	-.1331e-01	-.6955e-03
21	.2100e+00	.1225e+00	.1351e+00	-.1267e-01	-.6323e-03
22	.2200e+00	.1108e+00	.1228e+00	-.1204e-01	-.5748e-03
23	.2300e+00	.1003e+00	.1117e+00	-.1142e-01	-.5225e-03
24	.2400e+00	.9072e-01	.1015e+00	-.1081e-01	-.4750e-03
25	.2500e+00	.8208e-01	.9230e-01	-.1021e-01	-.4318e-03
26	.2600e+00	.7427e-01	.8391e-01	-.9632e-02	-.3926e-03
27	.2700e+00	.6721e-01	.7628e-01	-.9072e-02	-.3569e-03
28	.2800e+00	.6081e-01	.6934e-01	-.8533e-02	-.3244e-03
29	.2900e+00	.5502e-01	.6304e-01	-.8016e-02	-.2950e-03
30	.3000e+00	.4979e-01	.5731e-01	-.7521e-02	-.2681e-03
31	.3100e+00	.4505e-01	.5210e-01	-.7049e-02	-.2438e-03
32	.3200e+00	.4076e-01	.4736e-01	-.6600e-02	-.2216e-03
33	.3300e+00	.3688e-01	.4306e-01	-.6174e-02	-.2015e-03
34	.3400e+00	.3337e-01	.3914e-01	-.5769e-02	-.1831e-03
35	.3500e+00	.3020e-01	.3558e-01	-.5387e-02	-.1665e-03
36	.3600e+00	.2732e-01	.3235e-01	-.5025e-02	-.1514e-03
37	.3700e+00	.2472e-01	.2941e-01	-.4685e-02	-.1376e-03
38	.3800e+00	.2237e-01	.2673e-01	-.4364e-02	-.1251e-03
39	.3900e+00	.2024e-01	.2430e-01	-.4063e-02	-.1137e-03
40	.4000e+00	.1832e-01	.2209e-01	-.3779e-02	-.1034e-03
41	.4100e+00	.1657e-01	.2009e-01	-.3514e-02	-.9398e-04
42	.4200e+00	.1500e-01	.1826e-01	-.3265e-02	-.8544e-04
43	.4300e+00	.1357e-01	.1660e-01	-.3032e-02	-.7767e-04
44	.4400e+00	.1228e-01	.1509e-01	-.2814e-02	-.7061e-04
45	.4500e+00	.1111e-01	.1372e-01	-.2610e-02	-.6419e-04

46	.4600e+00	.1005e-01	.1247e-01	-.2420e-02	-.5835e-04
47	.4700e+00	.9095e-02	.1134e-01	-.2243e-02	-.5305e-04
48	.4800e+00	.8230e-02	.1031e-01	-.2078e-02	-.4823e-04
49	.4900e+00	.7447e-02	.9370e-02	-.1924e-02	-.4384e-04
50	.5000e+00	.6738e-02	.8519e-02	-.1781e-02	-.3986e-04
51	.5100e+00	.6097e-02	.7744e-02	-.1647e-02	-.3623e-04
52	.5200e+00	.5517e-02	.7040e-02	-.1524e-02	-.3294e-04
53	.5300e+00	.4992e-02	.6400e-02	-.1409e-02	-.2995e-04
54	.5400e+00	.4517e-02	.5818e-02	-.1302e-02	-.2722e-04
55	.5500e+00	.4087e-02	.5289e-02	-.1203e-02	-.2475e-04
56	.5600e+00	.3698e-02	.4809e-02	-.1111e-02	-.2250e-04
57	.5700e+00	.3346e-02	.4371e-02	-.1025e-02	-.2045e-04
58	.5800e+00	.3028e-02	.3974e-02	-.9464e-03	-.1859e-04
59	.5900e+00	.2739e-02	.3613e-02	-.8733e-03	-.1690e-04
60	.6000e+00	.2479e-02	.3284e-02	-.8055e-03	-.1537e-04
61	.6100e+00	.2243e-02	.2986e-02	-.7428e-03	-.1397e-04
62	.6200e+00	.2029e-02	.2714e-02	-.6848e-03	-.1270e-04
63	.6300e+00	.1836e-02	.2468e-02	-.6312e-03	-.1155e-04
64	.6400e+00	.1662e-02	.2243e-02	-.5816e-03	-.1050e-04
65	.6500e+00	.1503e-02	.2039e-02	-.5358e-03	-.9541e-05
66	.6600e+00	.1360e-02	.1854e-02	-.4935e-03	-.8674e-05
67	.6700e+00	.1231e-02	.1685e-02	-.4544e-03	-.7885e-05
68	.6800e+00	.1114e-02	.1532e-02	-.4184e-03	-.7169e-05
69	.6900e+00	.1008e-02	.1393e-02	-.3851e-03	-.6517e-05
70	.7000e+00	.9119e-03	.1266e-02	-.3543e-03	-.5924e-05
71	.7100e+00	.8251e-03	.1151e-02	-.3260e-03	-.5386e-05
72	.7200e+00	.7466e-03	.1046e-02	-.2999e-03	-.4896e-05
73	.7300e+00	.6755e-03	.9513e-03	-.2758e-03	-.4451e-05
74	.7400e+00	.6113e-03	.8649e-03	-.2536e-03	-.4046e-05
75	.7500e+00	.5531e-03	.7862e-03	-.2331e-03	-.3679e-05
76	.7600e+00	.5005e-03	.7148e-03	-.2143e-03	-.3344e-05
77	.7700e+00	.4528e-03	.6498e-03	-.1969e-03	-.3040e-05
78	.7800e+00	.4097e-03	.5907e-03	-.1810e-03	-.2764e-05
79	.7900e+00	.3707e-03	.5370e-03	-.1663e-03	-.2513e-05
80	.8000e+00	.3355e-03	.4882e-03	-.1527e-03	-.2284e-05
81	.8100e+00	.3035e-03	.4438e-03	-.1403e-03	-.2076e-05
82	.8200e+00	.2747e-03	.4035e-03	-.1288e-03	-.1888e-05
83	.8300e+00	.2485e-03	.3668e-03	-.1183e-03	-.1716e-05
84	.8400e+00	.2249e-03	.3334e-03	-.1086e-03	-.1560e-05
85	.8500e+00	.2035e-03	.3031e-03	-.9966e-04	-.1418e-05
86	.8600e+00	.1841e-03	.2756e-03	-.9146e-04	-.1289e-05
87	.8700e+00	.1666e-03	.2505e-03	-.8393e-04	-.1172e-05
88	.8800e+00	.1507e-03	.2277e-03	-.7701e-04	-.1066e-05
89	.8900e+00	.1364e-03	.2070e-03	-.7065e-04	-.9687e-06
90	.9000e+00	.1234e-03	.1882e-03	-.6481e-04	-.8806e-06
91	.9100e+00	.1117e-03	.1711e-03	-.5944e-04	-.8006e-06
92	.9200e+00	.1010e-03	.1556e-03	-.5451e-04	-.7278e-06
93	.9300e+00	.9142e-04	.1414e-03	-.4999e-04	-.6616e-06
94	.9400e+00	.8272e-04	.1286e-03	-.4583e-04	-.6015e-06
95	.9500e+00	.7485e-04	.1169e-03	-.4202e-04	-.5468e-06
96	.9600e+00	.6773e-04	.1062e-03	-.3851e-04	-.4971e-06
97	.9700e+00	.6128e-04	.9658e-04	-.3530e-04	-.4519e-06
98	.9800e+00	.5545e-04	.8780e-04	-.3235e-04	-.4108e-06
99	.9900e+00	.5017e-04	.7982e-04	-.2965e-04	-.3735e-06
100	.1000e+01	.4540e-04	.7257e-04	-.2717e-04	-.3395e-06

timestep control scheme, the corresponding results are shown in Table 6.3.

In both Table 6.2 and Table 6.3, the first column of data is the time point, the second column the exact solution, the third column the solution using the backward Euler formula, the fourth the global error and the fifth the estimated local truncation error. It can be seen that, by using the fixed timestep scheme, the order of global error varies from -1 to -4, and the order of local truncation error from -2 to -6; which indicates that in some time intervals the stepsize is too large and in other intervals the stepsize is unnecessarily small. On the other hand, by using the new timestep control scheme, the order of the global error and the local truncation error are kept within the range from -2 to -3 and the range from -3 to -4, respectively.

6.2.3. Conclusions

Dynamically varying the timestep is necessary for the timing analysis program to evaluate the simulated results accurately and efficiently. To ensure an accurate transient analysis, the timestep must be controlled to produce an acceptable amount of local truncation error at each timepoint. In this chapter, a new algorithm of timestep control is proposed, in which the next timestep is predicted using the LTE at the present timepoint and no timestep is rejected. The PREMOS program employs the Backward Euler integration with the LTE timestep control described in this chapter.

Table 6.3 Numerical Solution Obtained by Using New Timestep Control Scheme with $\lambda=-10$, $x_0=1$, epa=0.01 and epr=0.05.

	Time	Exact Sol.	Bk Euler	Glb Err.	LTE
0	.0000e+00	.1000e+01	.1000e+01	.0000e+00	.0000e+00
1	.5000e-02	.9512e+00	.9524e+00	-.1152e-02	-.1152e-02
2	.1000e-01	.9048e+00	.9070e+00	-.2192e-02	-.1097e-02
3	.1500e-01	.8607e+00	.8638e+00	-.3130e-02	-.1044e-02
4	.2000e-01	.8187e+00	.8227e+00	-.3972e-02	-.9947e-03
5	.2500e-01	.7788e+00	.7835e+00	-.4725e-02	-.9474e-03
6	.3000e-01	.7408e+00	.7462e+00	-.5397e-02	-.9023e-03
7	.3500e-01	.7047e+00	.7107e+00	-.5993e-02	-.8593e-03
8	.4000e-01	.6703e+00	.6768e+00	-.6519e-02	-.8184e-03
9	.4500e-01	.6376e+00	.6446e+00	-.6981e-02	-.7794e-03
10	.5000e-01	.6065e+00	.6139e+00	-.7383e-02	-.7423e-03
11	.5500e-01	.5769e+00	.5847e+00	-.7729e-02	-.7069e-03
12	.6000e-01	.5488e+00	.5568e+00	-.8026e-02	-.6733e-03
13	.6500e-01	.5220e+00	.5303e+00	-.8276e-02	-.6412e-03
14	.7000e-01	.4966e+00	.5051e+00	-.8483e-02	-.6107e-03
15	.7500e-01	.4724e+00	.4810e+00	-.8651e-02	-.5816e-03
16	.8000e-01	.4493e+00	.4581e+00	-.8783e-02	-.5539e-03
17	.8500e-01	.4274e+00	.4363e+00	-.8882e-02	-.5275e-03
18	.9000e-01	.4066e+00	.4155e+00	-.8951e-02	-.5024e-03
19	.9500e-01	.3867e+00	.3957e+00	-.8993e-02	-.4785e-03
20	.1000e+00	.3679e+00	.3769e+00	-.9010e-02	-.4557e-03
21	.1050e+00	.3499e+00	.3589e+00	-.9005e-02	-.4340e-03
22	.1100e+00	.3329e+00	.3418e+00	-.8979e-02	-.4133e-03
23	.1150e+00	.3166e+00	.3256e+00	-.8935e-02	-.3936e-03
24	.1200e+00	.3012e+00	.3101e+00	-.8874e-02	-.3749e-03
25	.1250e+00	.2865e+00	.2953e+00	-.8798e-02	-.3571e-03
26	.1300e+00	.2725e+00	.2812e+00	-.8709e-02	-.3400e-03
27	.1350e+00	.2592e+00	.2678e+00	-.8608e-02	-.3239e-03
28	.1400e+00	.2466e+00	.2551e+00	-.8497e-02	-.3084e-03
29	.1450e+00	.2346e+00	.2429e+00	-.8376e-02	-.2937e-03
30	.1500e+00	.2231e+00	.2314e+00	-.8247e-02	-.2798e-03
31	.1550e+00	.2122e+00	.2204e+00	-.8112e-02	-.2664e-03
32	.1600e+00	.2019e+00	.2099e+00	-.7970e-02	-.2538e-03
33	.1650e+00	.1920e+00	.1999e+00	-.7823e-02	-.2417e-03
34	.1700e+00	.1827e+00	.1904e+00	-.7671e-02	-.2302e-03
35	.1750e+00	.1738e+00	.1813e+00	-.7516e-02	-.2192e-03
36	.1800e+00	.1653e+00	.1727e+00	-.7359e-02	-.2088e-03
37	.1850e+00	.1572e+00	.1644e+00	-.7198e-02	-.1988e-03
38	.1900e+00	.1496e+00	.1566e+00	-.7037e-02	-.1894e-03
39	.1950e+00	.1423e+00	.1491e+00	-.6874e-02	-.1803e-03
40	.2000e+00	.1353e+00	.1420e+00	-.6710e-02	-.1717e-03

41	.2050e+00	.1287e+00	.1353e+00	-.6547e-02	-.1636e-03
42	.2100e+00	.1225e+00	.1288e+00	-.6383e-02	-.1558e-03
43	.2150e+00	.1165e+00	.1227e+00	-.6220e-02	-.1484e-03
44	.2200e+00	.1108e+00	.1169e+00	-.6058e-02	-.1413e-03
45	.2250e+00	.1054e+00	.1113e+00	-.5897e-02	-.1346e-03
46	.2300e+00	.1003e+00	.1060e+00	-.5738e-02	-.1282e-03
47	.2350e+00	.9537e-01	.1009e+00	-.5580e-02	-.1221e-03
48	.2400e+00	.9072e-01	.9614e-01	-.5424e-02	-.1162e-03
49	.2450e+00	.8629e-01	.9156e-01	-.5270e-02	-.1107e-03
50	.2500e+00	.8208e-01	.8720e-01	-.5119e-02	-.1054e-03
51	.2550e+00	.7808e-01	.8305e-01	-.4970e-02	-.1004e-03
52	.2600e+00	.7427e-01	.7910e-01	-.4823e-02	-.9564e-04
53	.2700e+00	.6721e-01	.7191e-01	-.4700e-02	-.3364e-03
54	.2800e+00	.6081e-01	.6537e-01	-.4559e-02	-.3059e-03
55	.2900e+00	.5502e-01	.5943e-01	-.4403e-02	-.2780e-03
56	.3000e+00	.4979e-01	.5402e-01	-.4237e-02	-.2528e-03
57	.3100e+00	.4505e-01	.4911e-01	-.4063e-02	-.2298e-03
58	.3200e+00	.4076e-01	.4465e-01	-.3886e-02	-.2089e-03
59	.3300e+00	.3688e-01	.4059e-01	-.3706e-02	-.1899e-03
60	.3400e+00	.3337e-01	.3690e-01	-.3526e-02	-.1726e-03
61	.3500e+00	.3020e-01	.3354e-01	-.3347e-02	-.1569e-03
62	.3600e+00	.2732e-01	.3050e-01	-.3171e-02	-.1427e-03
63	.3700e+00	.2472e-01	.2772e-01	-.2999e-02	-.1297e-03
64	.3800e+00	.2237e-01	.2520e-01	-.2832e-02	-.1179e-03
65	.3900e+00	.2024e-01	.2291e-01	-.2669e-02	-.1072e-03
66	.4000e+00	.1832e-01	.2083e-01	-.2513e-02	-.9745e-04
67	.4100e+00	.1657e-01	.1894e-01	-.2362e-02	-.8859e-04
68	.4200e+00	.1500e-01	.1721e-01	-.2218e-02	-.8054e-04
69	.4300e+00	.1357e-01	.1565e-01	-.2080e-02	-.7322e-04
70	.4400e+00	.1228e-01	.1423e-01	-.1949e-02	-.6656e-04
71	.4500e+00	.1111e-01	.1293e-01	-.1824e-02	-.6051e-04
72	.4700e+00	.9095e-02	.1078e-01	-.1682e-02	-.1889e-03
73	.4900e+00	.7447e-02	.8981e-02	-.1535e-02	-.1574e-03
74	.5100e+00	.6097e-02	.7484e-02	-.1388e-02	-.1311e-03
75	.5300e+00	.4992e-02	.6237e-02	-.1245e-02	-.1093e-03
76	.5500e+00	.4087e-02	.5197e-02	-.1111e-02	-.9108e-04
77	.5700e+00	.3346e-02	.4331e-02	-.9852e-03	-.7590e-04
78	.6100e+00	.2243e-02	.3094e-02	-.8508e-03	-.1904e-03
79	.6500e+00	.1503e-02	.2210e-02	-.7064e-03	-.1360e-03
80	.7300e+00	.6755e-03	.1228e-02	-.5521e-03	-.2347e-03
81	.8100e+00	.3035e-03	.6820e-03	-.3785e-03	-.1304e-03
82	.9700e+00	.6128e-04	.2623e-03	-.2010e-03	-.1246e-03
83	.1000e+01	.4540e-04	.2018e-03	-.1564e-03	-.7453e-05

CHAPTER 7

The PREMOS Program

7.1. Introduction

PREMOS (PREdiction-based simulator for MOS circuits) is an experimental simulator program for VLSI MOS digital circuits. The object of the program is to close the gap between conventional circuit simulation and logic simulation. This program is faster than conventional circuit simulators because it uses a Gauss-Seidel circuit simulation scheme and employs built-in models for the subcircuits. Although it is slower than logic simulators, it generates more accurate electrical waveforms than the logic levels produced by logic simulators.

In PREMOS, a modified block Gauss-Seidel-Newton algorithm is used instead of the standard point Gauss-Jacobi algorithm used in MOTIS and MOTIS-C. Compared with MOTIS and MOTIS-C, the accuracy of the results is improved at three levels: (1) circuit analysis is used to solve the unilateral subcircuit equations, (2) multi-nonlinear iteration is used at each time point, and (3) the predictor method is used for solving the feedback interdependence. In addition, an analysis sequencing algorithm based on relevant parts is used to improve the speed of the simulation. Because of the additional iterations, PREMOS is generally about five times slower than that of MOTIS-C, whereas the speed and circuit-size capability of MOTIS-C

have been claimed to be over two orders of magnitude greater than those of SPICE2 [7].

PREMOS evolves from MOTIS-C but has different data structures and new analysis algorithms. It is written in FORTRAN and contains more than 3000 statements at the present time.

7.2. The Input Processing

The input processor reads and processes the model file of MOS devices, and the input data file containing the description of circuit elements and control statements. There are a number of built-in models for the partitioned subcircuits in this program. A list of circuit elements and their corresponding models is shown in Appendix 2. The control statements are listed in Appendix 3. The input processor constructs the internal node table and the linked lists for the structure of the circuit. The data structures for subcircuit models are shown in Appendix 4. The data generated are passed to the analysis part of the program through disc files.

The subroutine EROR performs error checking when input data are read. If an error exists, the program stops with the error messages printed out.

7.3. The Analysis Core

The analysis core of the program includes two phases: analysis sequencing phase and analysis phase. They are described respectively in this section.

7.3.1. Analysis Sequencing Phase

Before the transient analysis is performed, the sequence of analyzing the subcircuits is generated in this phase. First, the linked list of the corresponding directed graph is constructed. Algorithm 5 mentioned in Chapter 3 is then executed to provide the analysis sequence. During the sequencing procedures, the feedback paths and the floating capacitors are identified. Finally, the subcircuits that do not belong to the relevant set are deleted from the sequence.

7.3.2. Analysis Phase

Following the analysis sequencing, each scheduled subcircuit is identified and linked to its models. Initially, device sizes and node tables are read out by means of the pointers in the model map. If feedback paths or floating capacitors exist, the associated node voltages are predicted. If a subcircuit has been declared latent at a previous timepoint and its fan-in node voltage changes are less than some certain limit, the subcircuit is bypassed during the analysis. At each nonlinear iteration, the device models are evaluated and the subcircuit matrix is formed. In solving the

matrix, no sparse matrix technique and no reordering scheme in the LU factorization process are used since the size of the subcircuit is usually small. The number of nonlinear iterations is specified by the user. The timestep can be either controlled by local truncation error as described in Chapter 6 or fixed, depending on the user's option.

7.4. The Output Processing

The output from the program is written in a disc file, which could be read by the output processor. The output data is sent to either the line printer or the plotting terminal through the output processor. Hard copies of the plots of the waveforms selected can also be produced on an X-Y plotter.

7.5. Simulated Examples

In this section, four examples of circuit simulations using PREMOS are presented. Example 1 illustrates the timing analysis of a PLA circuit. The input data file is also included. Example 2 and Example 3 are given to show the improvements in the accuracy of the simulated results by using the predictor method. The comparisons with SPICE2 and MOTIS-C are also included in these two examples. Example 4 shows the effect of analyzing only the relevant parts on reducing the simulation time.

7.5.1. PLA Circuit

The circuit diagram of a programmable logic array (PLA), which is used to implement a traffic light controller, is shown in Fig. 7.1 [39]. This PLA circuit which is composed of about 150 MOS transistors can be partitioned into 42 unilateral subcircuits or circuit elements. The input data file used for circuit simulation by PREMOS, is included in Appendix 5.

The input and output waveforms for this example simulated by PREMOS are shown in Fig. 7.2. The total analysis time is 12.450 seconds, compared with 5.567 seconds taken by MOTIS-C. Table 7.1, which can be found in [39], is given to verify the simulated results.

7.5.2. Bootstrap Capacitor Circuit

The bootstrap capacitor circuit in Fig. 7.3 has become very popular in MOS digital circuit design for fast switching operations and large driving capability. The simulated results are shown in Fig. 7.4, where it can be seen that the modified Gauss-Seidel method used in PREMOS is more accurate than the standard Gauss-Seidel method (represented by PREMOS without predictor) and the Gauss-Jacobi method used in MOTIS-C. In this comparison the exact solution is what SPICE2 produced.

In this example, the analysis time is 17.60 seconds for SPICE2, 1.017 seconds for PREMOS with predictor, 1.200 seconds for PREMOS without predictor and 0.467 seconds for MOTIS-C. Both PREMOS and MOTIS-C use a fixed timestep scheme with a 0.5 ns timestep. PREMOS

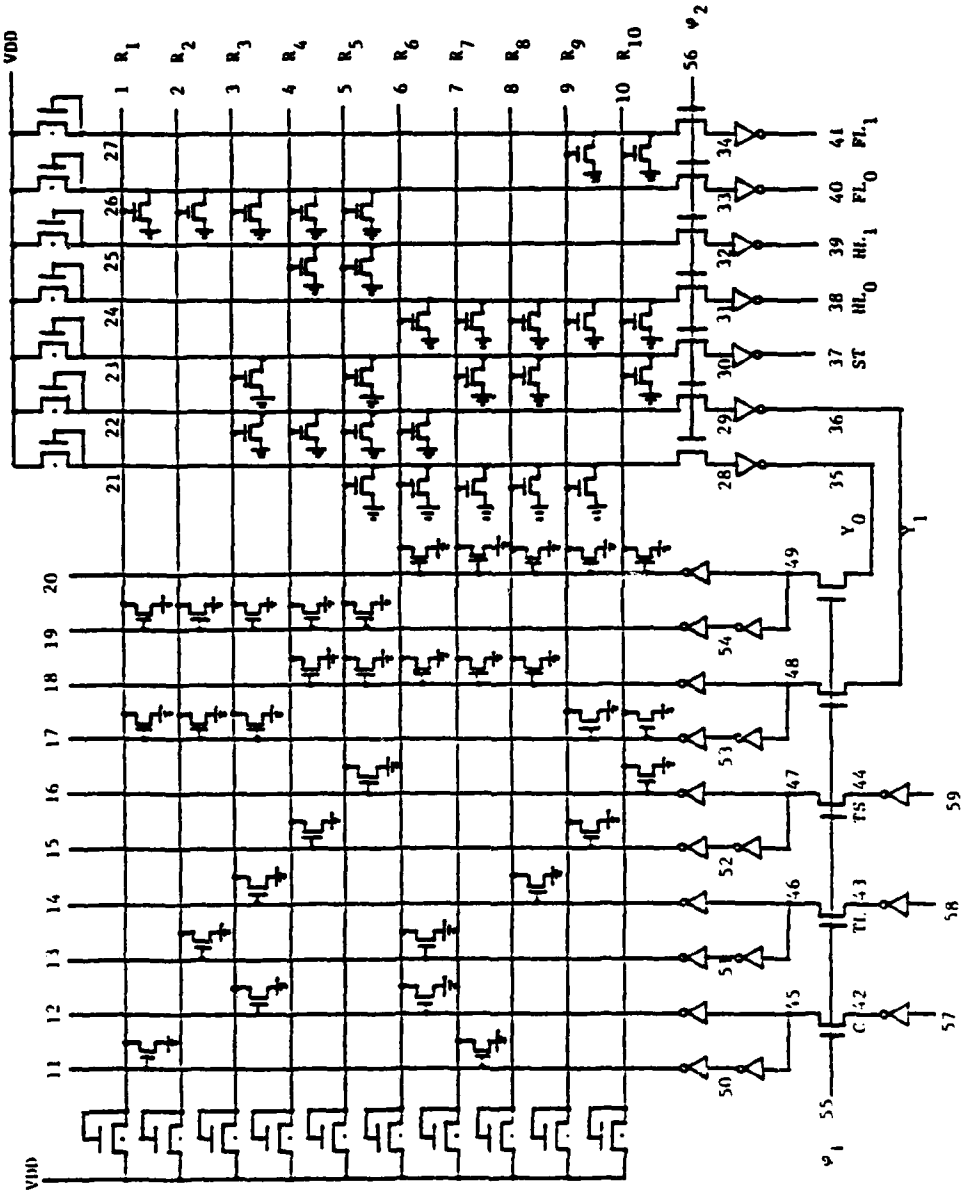


Fig. 7.1 Circuit Diagram of a PLA Example.

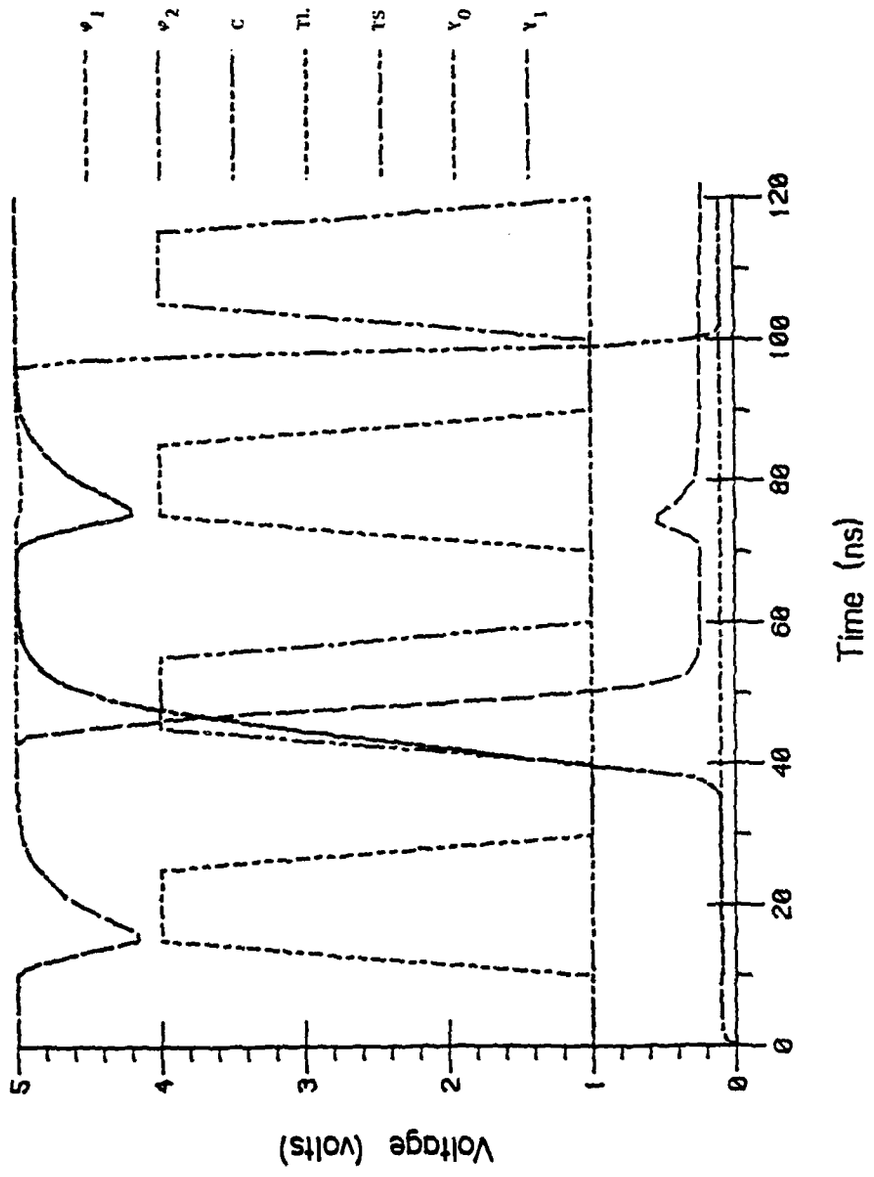


Fig. 7.2 (a) Voltage Waveforms φ_1, φ_2, C, TL, TS, Y_0, and Y_1 for the PLA Circuit in Fig. 7.1.

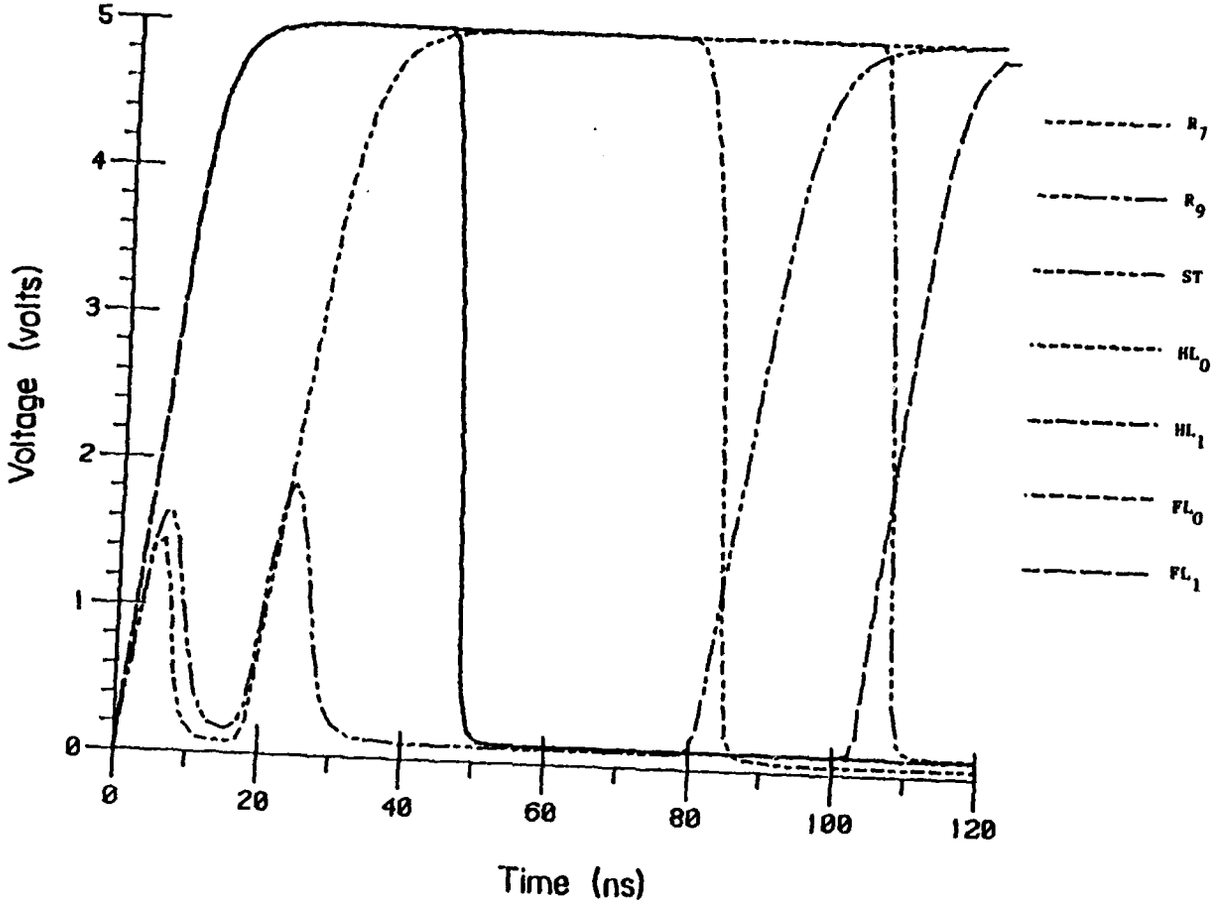


Fig. 7.2 (b) Output Waveforms R_7, R_9, ST, HL_0, HL_1, FL_0 and FL_1 for the PLA Circuit in Fig. 7.1.

Table 7.1 Encoded State Transition Table for the Light Controller.

Stored during φ_1 in In-register				Stored during φ_2 in Out-register						
Inputs			Present state	Next state	Outputs					Product terms
C	TL	TS	Y_{n0}, Y_{n1}	Y_{n0}, Y_{n1}	ST	HL_0	HL_1	FL_0	FL_1	
0	X	X	0.0(HG)	0.0(HG)	0	0	0	1	0	R_1
X	0	X	0.0(HG)	0.0(HG)	0	0	0	1	0	R_2
1	1	X	0.0(HG)	0.1(HY)	1	0	0	1	0	R_3
X	X	0	0.1(HY)	0.1(HY)	0	0	1	1	0	R_4
X	X	1	0.1(HY)	1.1(FG)	1	0	1	1	0	R_5
1	0	X	1.1(FG)	1.1(FG)	0	1	0	0	0	R_6
0	X	X	1.1(FG)	1.0(FY)	1	1	0	0	0	R_7
X	1	X	1.1(FG)	1.0(FY)	1	1	0	0	0	R_8
X	X	0	1.0(FY)	1.0(FY)	0	1	0	0	1	R_9
X	X	1	1.0(FY)	0.0(HG)	1	1	0	0	1	R_{10}

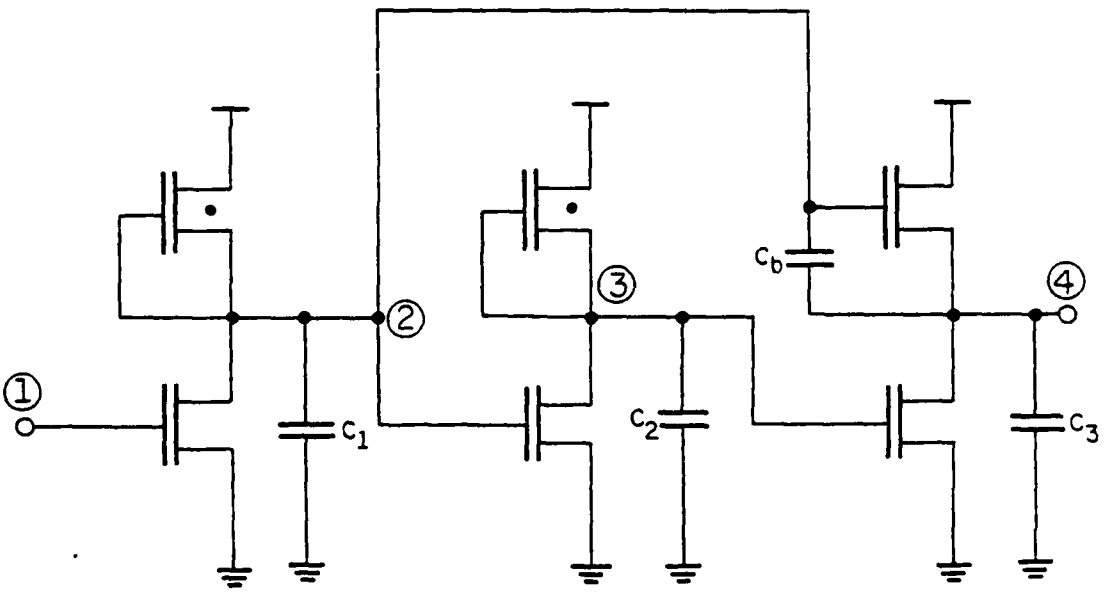


Fig. 7.3 Bootstrap Capacitor Circuit.

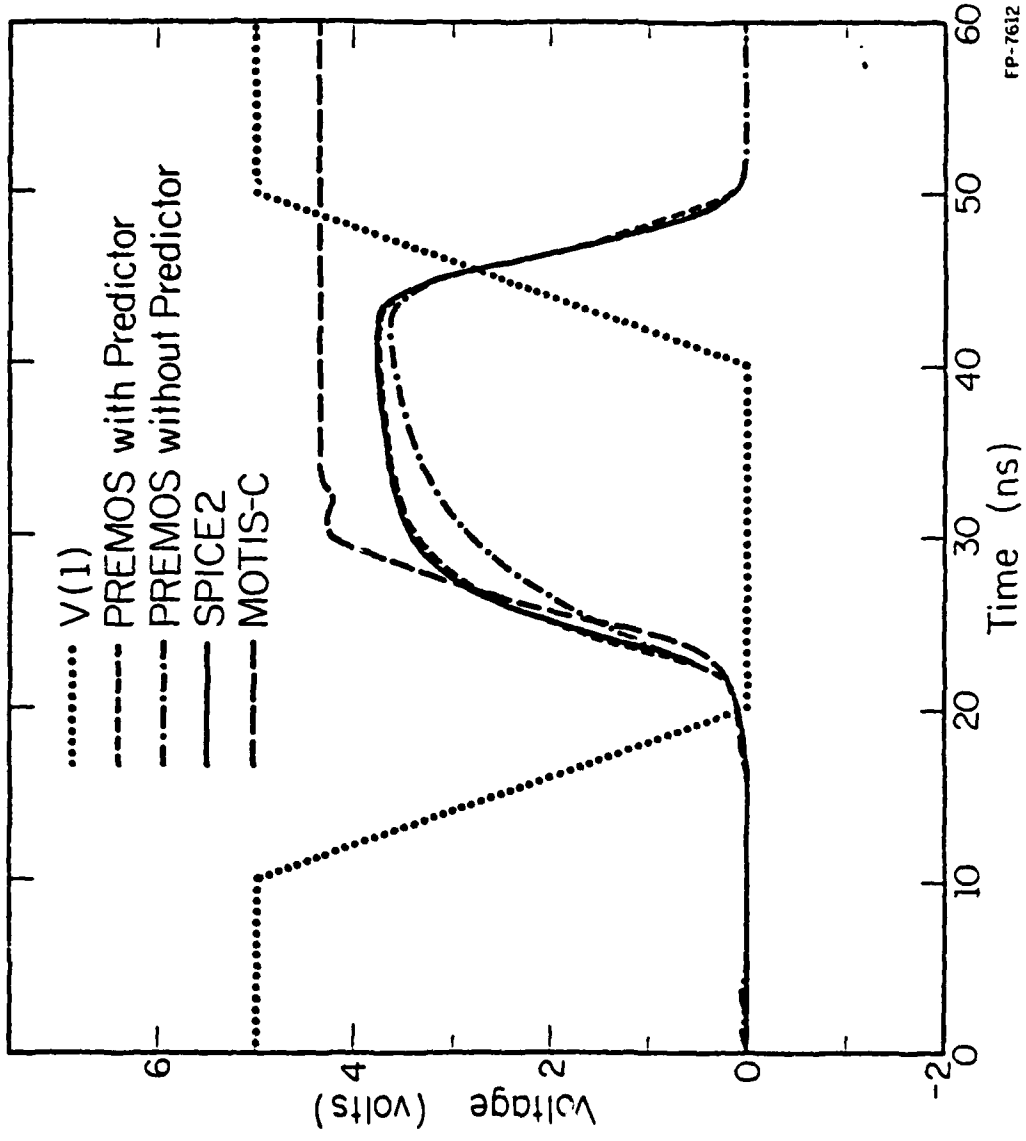


Fig. 7.4 Responses for the Circuit in Fig. 7.3.

uses three nonlinear iterations at each timepoint.

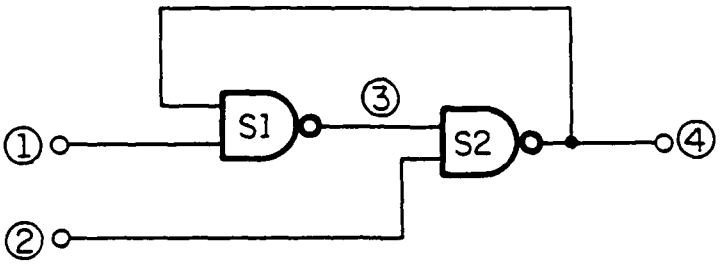
7.5.3. One-Bit Register

The block diagram and circuit schematic of a one-bit register circuit are shown in Fig. 7.5. This design has a memory function. A feedback path exists from the output of S2 to the input of S1. As shown in Fig. 7.6, by comparing the results to SPICE2 results, it can be seen that PREMOS with predictor produces more accurate results than both PREMOS without predictor and MOTIS-C.

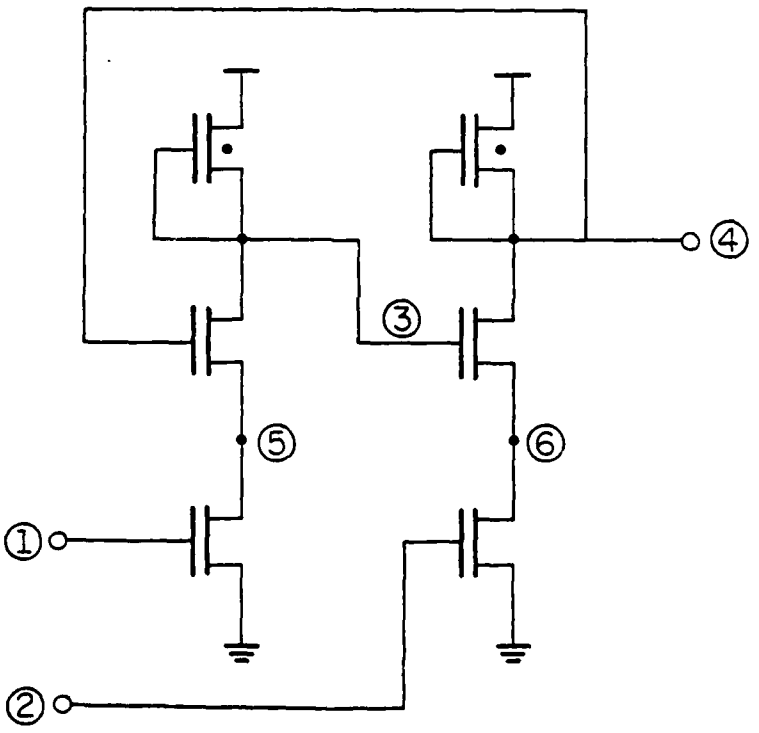
In this example, the analysis time is 19.05 seconds for SPICE2, 1.267 seconds for PREMOS with predictor, 1.000 seconds for PREMOS without predictor and 0.433 seconds for MOTIS-C. Both PREMOS and MOTIS-C employ a fixed timestep of 0.5 nanoseconds in the transient analysis. In PREMOS three nonlinear iterations are used at each timepoint.

7.5.4. Binary-to-Octal Decoder

A block diagram of the binary-to-octal decoder circuit is shown in Fig. 7.7. In this example only the subcircuits that directly or indirectly affect the output are analyzed during the solution process. The analysis results of selecting one, two, four or eight outputs are shown in Table 7.2. Because of the overhead involved in the simulation, the total CPU time taken in each case is not in proportion to the number of relevant subcircuits.



(a)



(b)

Fig. 7.5 (a) Block Diagram
(b) Circuit Schematic of One-Bit Register.

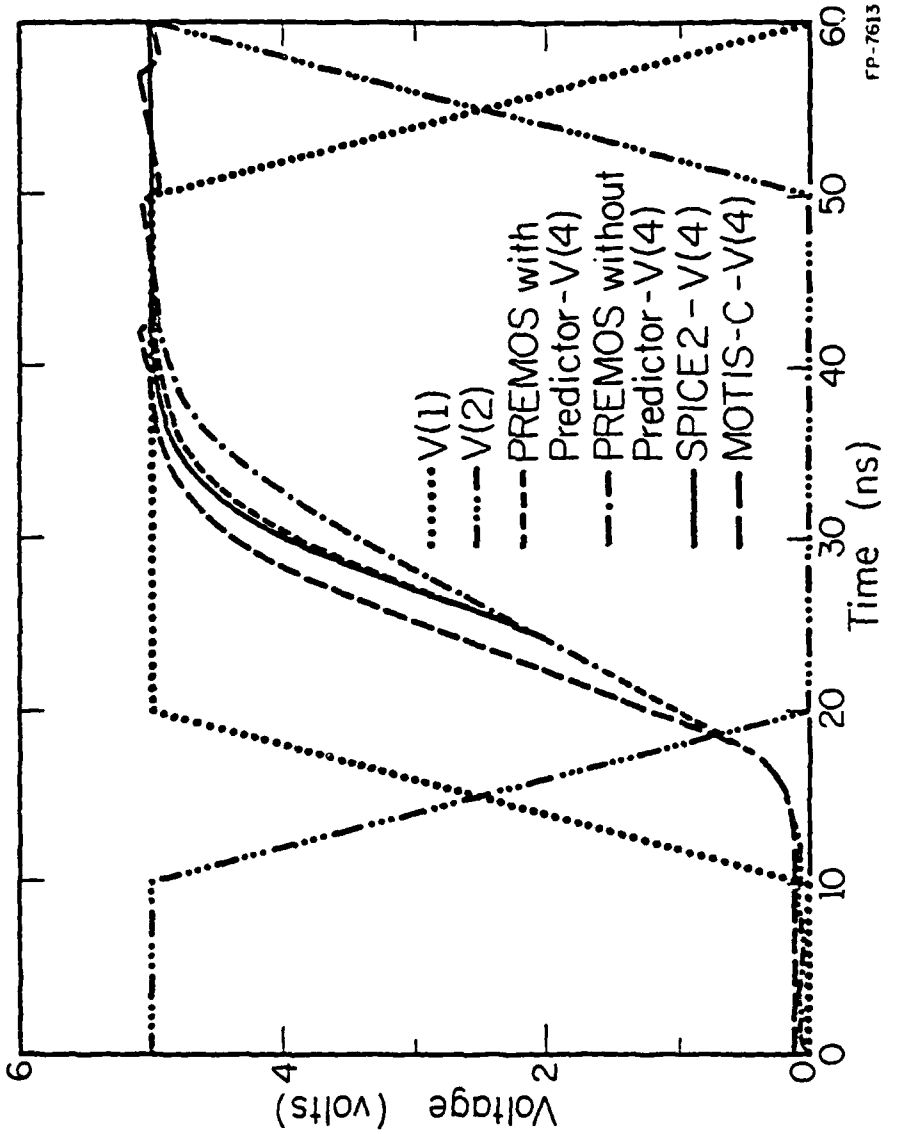


Fig. 7.6 Responses for the Circuit in Fig. 7.5.

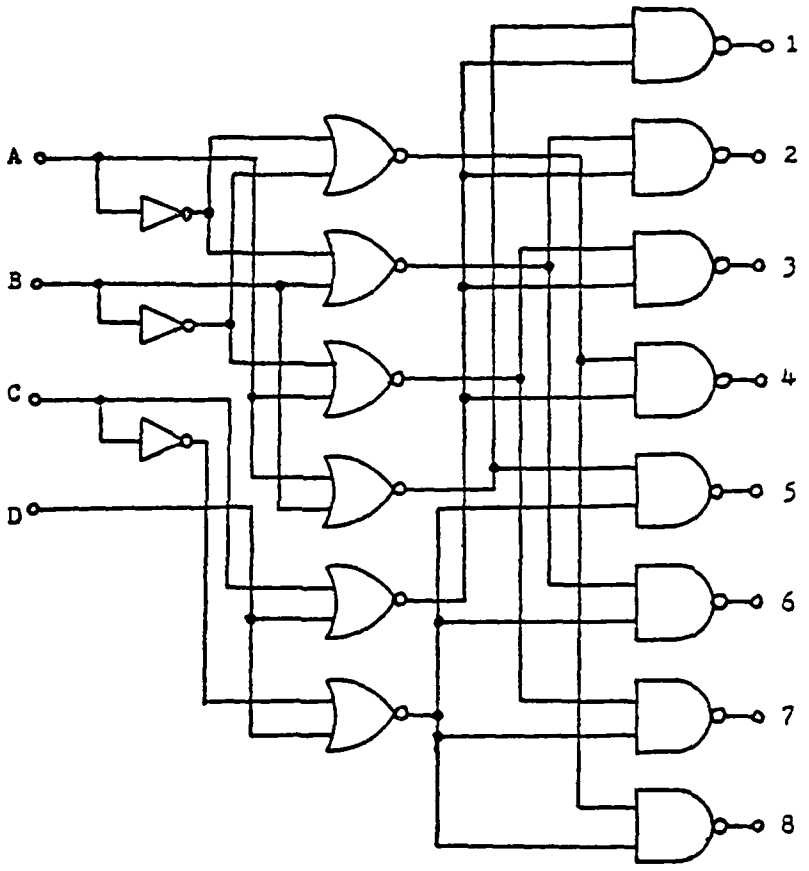


Fig. 7.7. Block Diagram of Binary-to-Octal Decoder.

Table 7.2 Simulation Data for the Circuit in Fig. 7.7.

Output Requested	No. of Relevant Subcircuits	Analysis Time (seconds)
1 2 3 4 5 6 7 8	17	11.167
1 2 3 4	11	8.933
5 6 7 8	12	9.200
1 2	6	6.817
3 4	7	7.183
5 6	7	7.100
7 8	7	7.133
1	3	5.333
2	4	5.817
3	4	5.800
4	5	5.883
5	4	5.567
6	5	5.967
7	5	5.683
8	6	6.650

CHAPTER 8

Conclusions

The aim of large-scale circuit simulation is to bridge the gap between conventional circuit simulation and logic simulation. In the experimental program PREMOS developed as part of this dissertation, the subcircuits are analyzed at the transistor level by using Newton's method as is done in conventional circuit simulation; but the signal propagation from subcircuit to subcircuit, which determines the analysis sequence of these unilateral subcircuits, is similiar to that used in logic simulation. The transistor level simulation in the subcircuits provides the detailed waveforms. The analysis sequencing combined with latency checking reduces significantly the amount of computation time and memory requirements.

The analysis sequencing procedure which includes checking and identifying feedback loops has been presented in Chapter 3. The procedure schedules only those subcircuits that directly or indirectly affect the output. Combined with latency checking, this 'segmentation' approach achieves a further increase in speed. The amount of increase depends on the circuit being analyzed.

In Chapter 4 we discuss initial DC analysis in large-scale circuit simulation and compare the different algorithms using 2-element and 3-element companion models for the MOS transistors. It is found that the 2-element model of the MOS transistor is suitable for

large-scale circuit simulation from the point of convergence rate. The analysis results also show that using a small fixed number of iterations produces DC solutions close enough to those obtained after much more iterations, provided the initial guess is a good approximation.

In using the standard Gauss-Seidel method for solving the partitioned circuit, the feedback loop is decoupled by assuming that there is no change in the feedback loop over the integration timestep. In this thesis, a 'modified' Gauss-Seidel method is proposed, where explicit formulas are used to predict the 'unsolved' variables in the feedback loops. As a result, the accuracy is improved without requiring much additional computation. It has been shown that the method is consistent, stable and convergent. It has also been shown that no parasitic oscillatory component appears in the solution if the timestep is smaller than a critical timestep.

As the entire circuit is partitioned into 'one-way' subcircuits, which can be easily identified as an 'event' during the simulation, latency detection and exploitation is used to provide additional computational savings. The latency criterion for PREMOS is described in Chapter 6. In the same chapter, a timestep control scheme based on the local truncation error is discussed. It should be noted that the proposed timestep control scheme does not reject the present timestep even when the LTE bound is exceeded.

The program PREMOS is described in some detail in Chapter 7. It is written for use on VAX 780/11 UNIX operating system. Several

simulation examples are given to show the validity of using the new algorithms and schemes.

PREMOS could be used for the timing analysis of MOS integrated circuits in hierarchical design. For general purpose usage, more enhancements on the program must be done. The input processor should be able to expand the macro or nested subcircuits. Furthermore, the capability of partitioning the circuit automatically should be added. In this way, the program could be used for verifying the circuit extracted from the layout. For other IC technologies like CMOS and I^2L, which could have unilateral gates (subcircuits) formed easily, the analysis techniques used in PREMOS can be applied to develop similiar kinds of simulators.

As described in [23], the speed of logic simulation could be increased more than several hundred times by using logic processors and array processors. Similiarly, it could be possible to implement decomposition algorithms in hardware and analyze 'one-way' subcircuits by using multi-processors to gain orders of magnitude in execution time for the next generation circuit simulators [12]. Further research on this kind of simulation machine could be promising in the future.

Appendix 1

MOS Device Modeling and Capacitor Modeling

In conventional circuit analysis, the MOS device model shown in Fig. A1.1(a) is generally used (the charge storage effects are not shown here). The three values of g_m, g_{ds} and i_o are calculated for each DC iteration. The substrate bias effect is included into the changes of the threshold voltages. For large-scale circuit analysis, the MOS device model for the enhancement transistor could be simplified further as shown in Fig. A1.1(b). Only two values, g_{ds} and i_o, need to be calculated at each DC iteration.

Recently, short-channel MOS devices are becoming widely used and "second-order" effects such as mobility reduction, channel length modulation and substrate bias effects are becoming increasingly important in deriving device models. In order to obtain accurate simulated results, these effects should be taken into account. The MOS device modeling work in PREMOS is based on curve-fitting empirical curves of DC characteristics, with the emphasis on matching the saturated points and the conductances in the saturated regions for different V_{GS}.

The modeling equations are

$$I_{DS} = KP * \frac{1 + \lambda V_{DS}}{1 + \eta(V_{GS} - V_T)} * ((V_{GS} - V_T)V_{DS} - \frac{1}{2} V_{DS}^2) \qquad (A1.1)$$

for operation in linear region

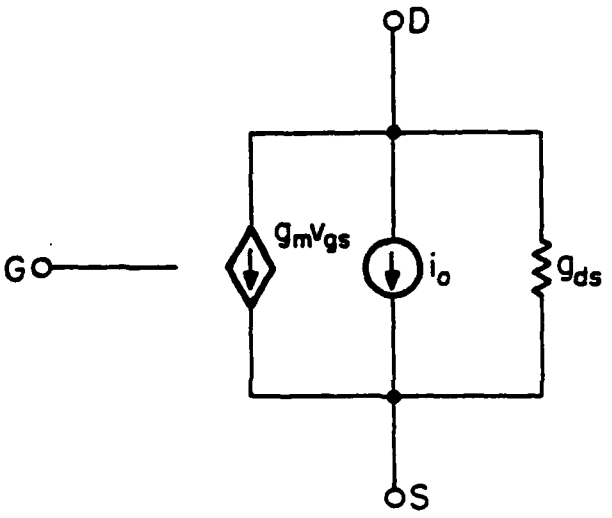


Fig. A1.1 (a) MOS Transistor Model in Conventional Circuit Analysis.

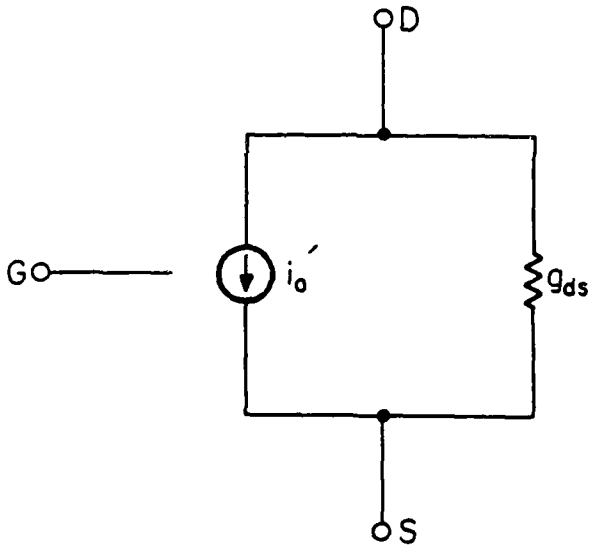


Fig. A1.1 (b) MOS Transistor Model in Large-Scale Circuit Analysis.

$$I_{DS} = KP * \frac{1 + \lambda V_{DS}}{1 + \eta(V_{GS} - V_T)} * (V_{GS} - V_T)^2 \qquad (A1.2)$$

for operation in saturation region and

$$V_T = V_{T0} + \Delta V_T \qquad (A1.3)$$

where

KP = intrinsic transconductance ($= \mu_o c_o W/L_{eff}$)

λ = channel length modulation parameter

η = mobility reduction parameter

V_{T0} = threshold voltages where the DC curves are measured

ΔV_T = threshold voltage change from V_{T0} due to substrate bias voltage change

ΔV_T is represented in tabular form as a function of the source-to-substrate voltage V_{SB}.

The extraction of the DC model parameters from the physical device can be done by using curve-fitting techniques in a straightforward manner. A special computer program can be developed for this purpose.

The capacitance at each node in LSI or VLSI circuits consists of two types (1) voltage-dependent capacitance formed by MOS devices, which includes gate capacitance and diffusion capacitance and (2) the interconnect capacitance. In the scaling down technology, the latter plays an increasingly important role in the circuit behavior. The features and modeling of these capacitances are described below:

(1) The voltage-dependent capacitance: gate capacitance and diffusion capacitance.

The voltage-dependent relation of gate capacitance for MOS transistor is shown in Fig. A1.2 [40]. For the diffusion capacitance c, the voltage-dependent relation can be expressed as

$$c = \frac{c_{jo}}{(1 - v/v_{bi})^{e}} \qquad \text{(A1.4)}$$

where

c_{jo} = diffusion capacitance at zero junction voltage

v = junction voltage

v_{bi} = junction contact potential

e = grading constant

In large-scale circuit simulation, it is rather expensive to calculate these voltage-dependent capacitances at every iteration. So these capacitances will be lumped together and approximated by a linear capacitance. The value of this linear capacitance depends on the device size and on the processing parameters. This value could be experimentally determined as a function of device size and gate oxide capacitance.

(2) Modeling of the interconnection capacitance.

Poly and metal capacitances are usually calculated by applying the parallel-plate formula. However, present scaling-down technology produces interconnection conductors that are comparable in dimension to the thickness of the oxide, so the parasitic capacitance of various interconnections can no longer be treated as capacitance of infinite parallel plates because of fringing field effects. For today's interconnection system, an error of a factor of two could

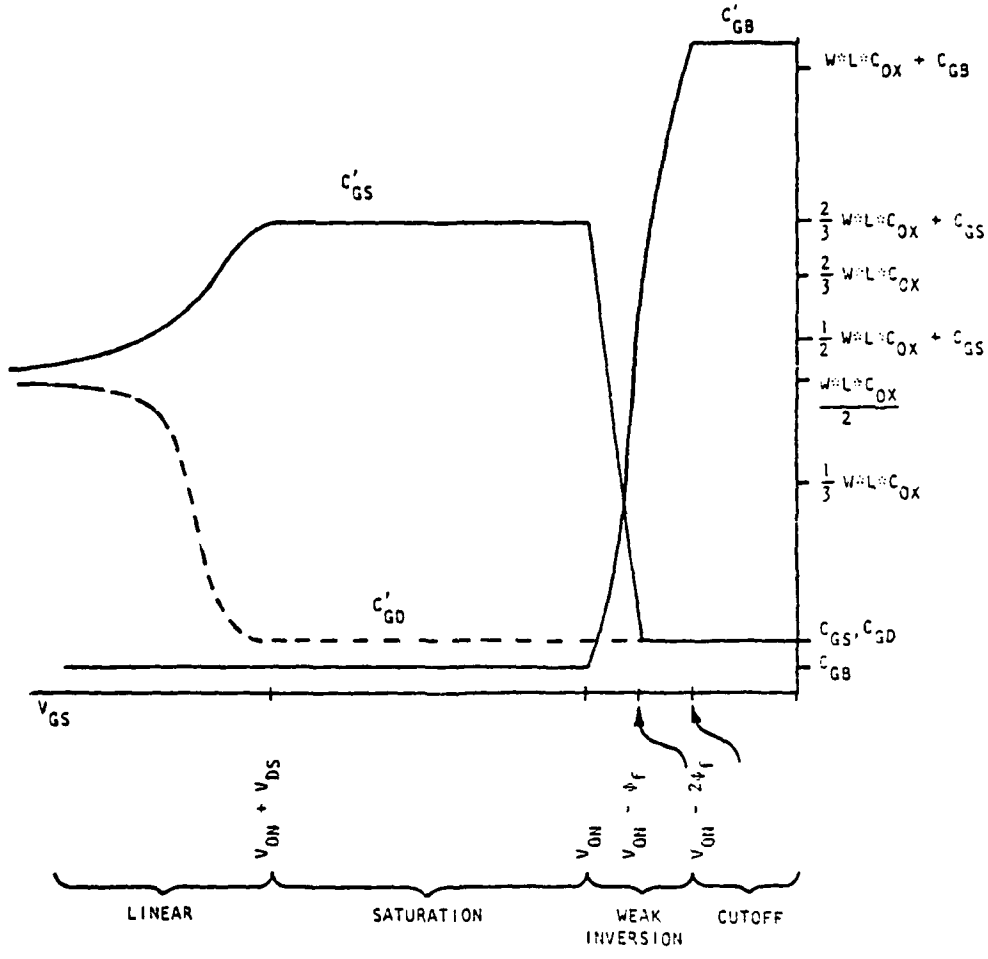


Fig. A1.2 Gate Capacitances of MOS Transistor.

result due to fringing fields alone! It is necessary then to correct the capacitance from the parallel-plate formula by a correction factor. The correction factor c/c_o could be evaluated either experimentally with test chips or theorectically, where

c = the interconnect wiring parasitic capacitance per unit length

c_o = the capacitance per unit length from the parallel-plate formula [41, 42].

Thus, the magnitude of the interconnection capacitance c_{intcon} can be obtained from

$$c_{intcon} = c_o * L * (\text{Correction Factor}) \qquad (A1.5)$$

Appendix 2

Input Descriptions for Circuit Elements and Their Models

The following types of subcircuit models have been implemented in the program PREMOS:

NAND2: 2-input NAND gate (Fig. A2.1)

NOR2 : 2-input NOR gate (Fig. A2.2)

ANDOI: n-input AND-OR-Inverter (Fig. A2.3)

ORANI: n-input OR-AND-Inverter (Fig. A2.4)

TRANS: n-input NOR gate with nt Transfer Gates (Fig. A2.5)

TRANP: same as TRANS except node tnt is taken as input

PUSPL: Push-Pull Inverter (Fig. A2.6)

LATCH: Latch Gate (Fig. A2.7)

SOURC: Clock (Voltage Source) Model

Model is described as

MODEL (mdnam) (type) (parameters)

for example,

MODEL m1 NAND2 (1 0.2 10f 20f 100f)

MODEL is the keyword for model description. mdnam is a user-defined name for the model. The type field specifies the model type. The available model types and their associated model parameters are listed in the following table:

TYPE	PARAMETERS
NAND2	wla wll ca ci cl

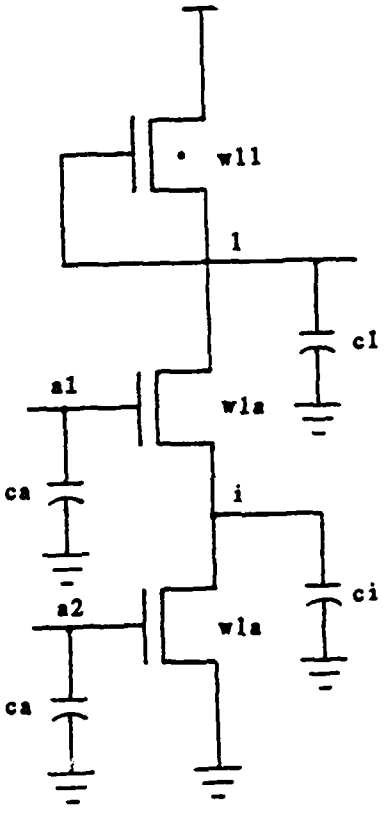


Fig. A2.1 2-Input NAND Gate.

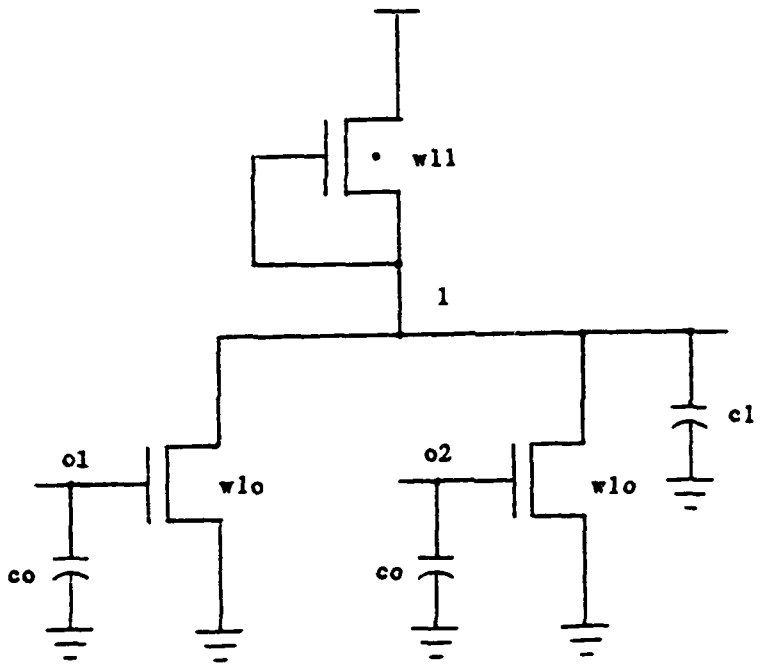


Fig. A2.2 2-Input NOR Gate.

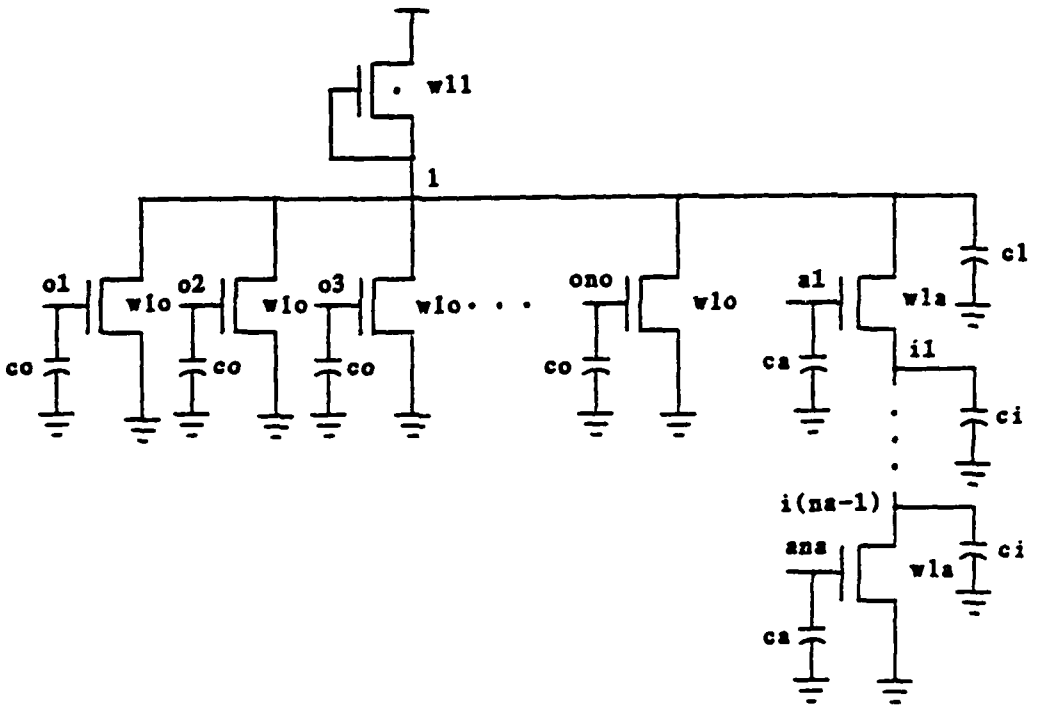


Fig. A2.3 n-Input AND-OR-Inverter.

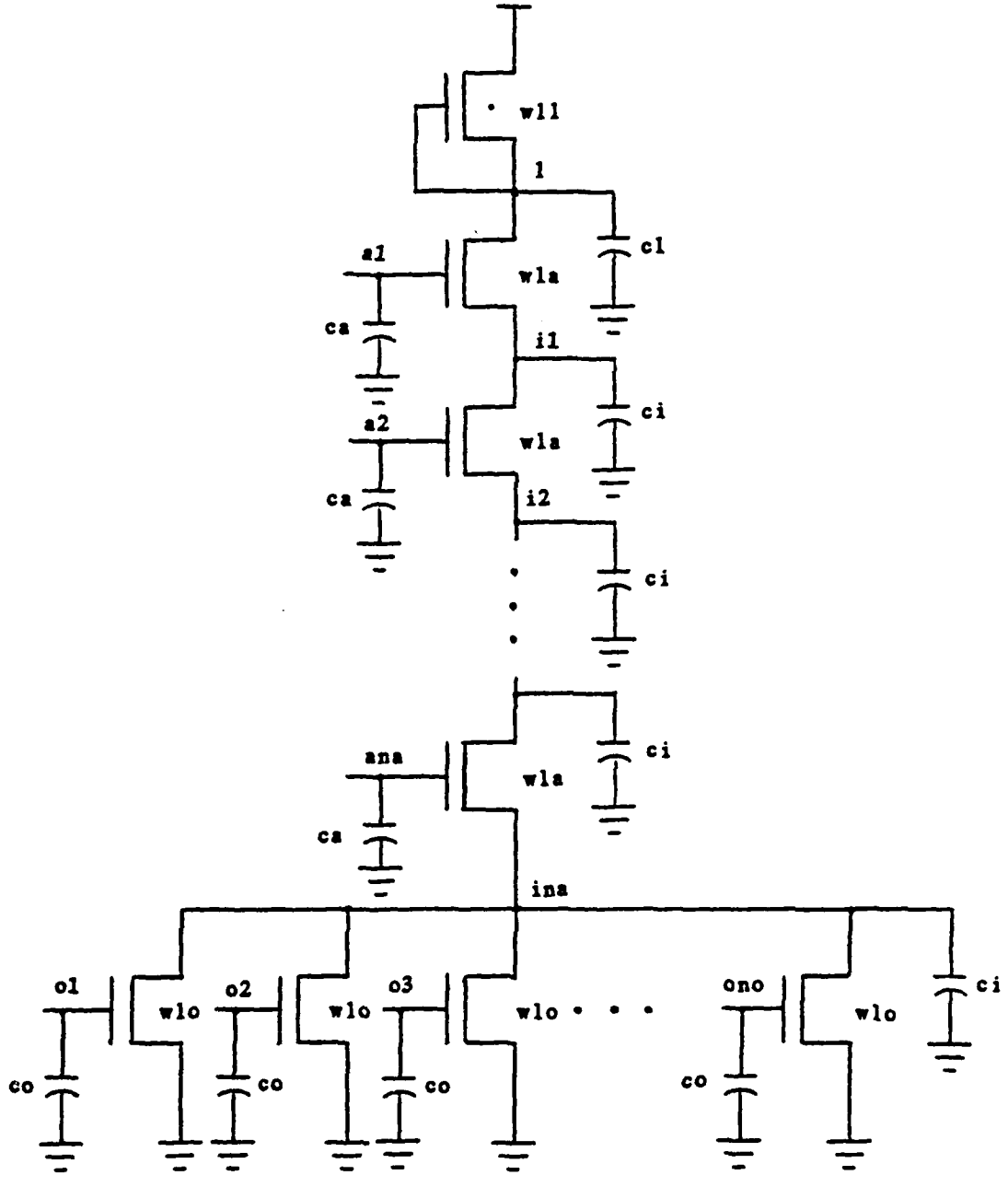


Fig. A2.4 n-Input OR-AND-Inverter.

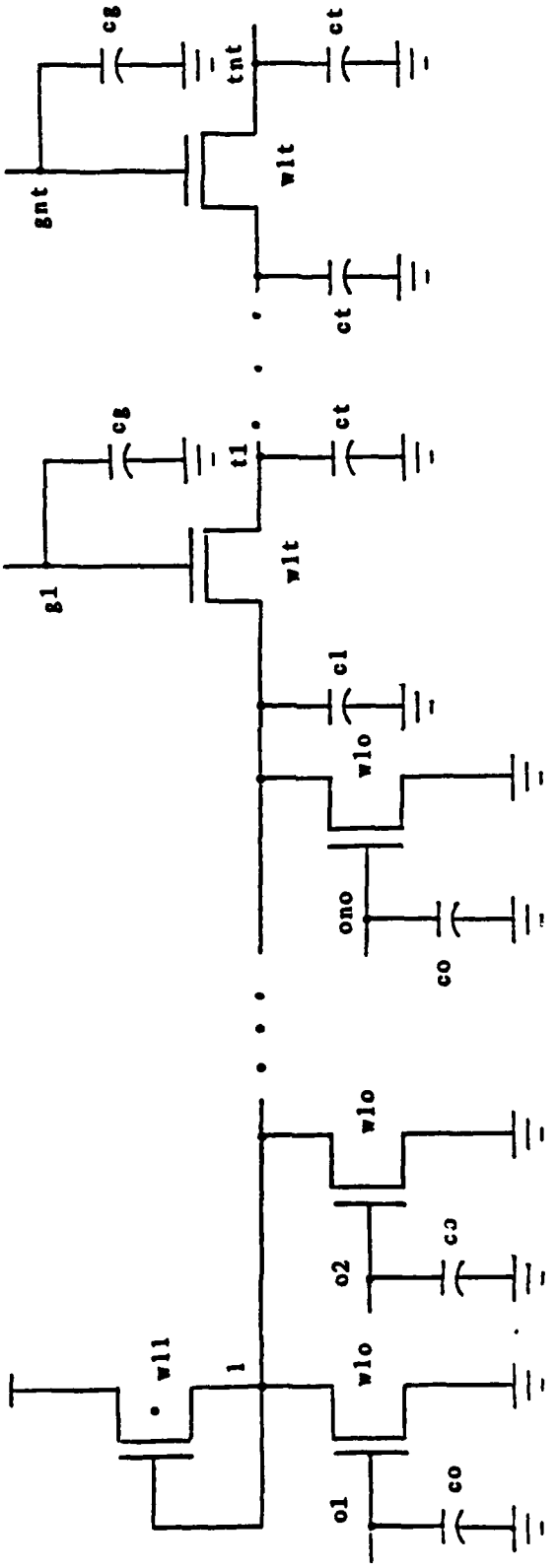


Fig. A2.5 n-Input NOR Gate with nt Transfer Gates.

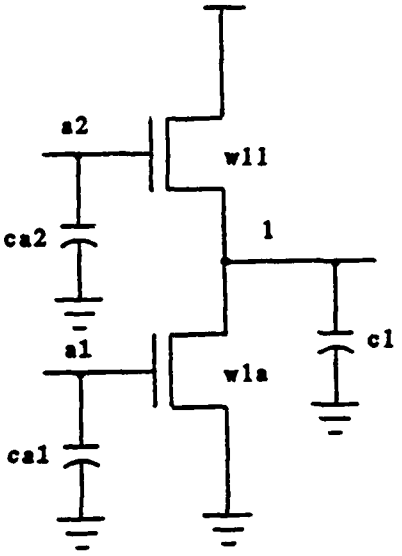


Fig. A2.6 Push-Pull Inverter.

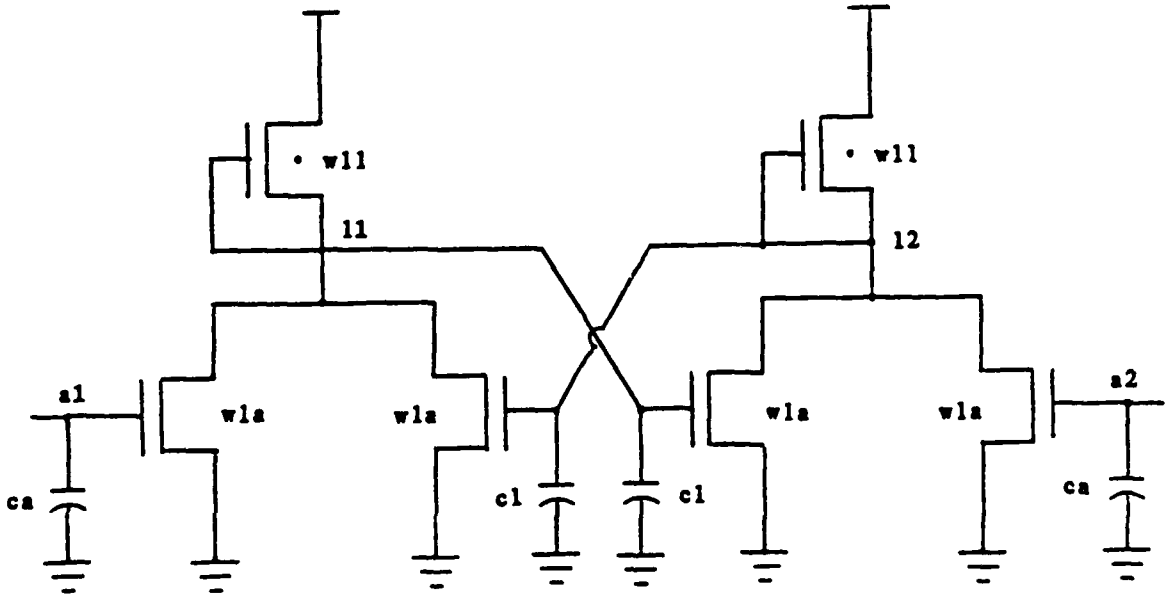


Fig. A2.7 Latch Gate.

```
NOR2      wlo wl1 co c1
ANDOI     wla wlo wl1 ca co ci c1 na no
ORANI     wlo wla wl1 co ca ci c1 no na
TRANS     wlo wl1 wlt co c1 cg ct no nt
TRANP     wlo wl1 wlt co c1 cg ct no nt
PUSPL     wla wl1 ca1 ca2 c1
LATCH     wla wl1 ca c1
SOURC     v1 v0 t0 tr t1 tf
```

The circuit is described as

```
(name) (nodes) (mdnam)
```

for example,

```
N1 1 2 3 4 NAND2
```

"name" is the name of the circuit element. The nodes field contains the node numbers describing the circuit connection. The order of the node numbers for each type of the subcircuit is listed in the table below:

```
TYPE      ORDER OF NODE NUMBERS
NAND2     a1 a2 i 1
NOR2      o1 o2 1
ANDOI     a1 a2 ... o1 o2 ... 1 i1 i2 ... i(na-1)
ORANI     o1 o2 ... a1 a2 ... 1 i1 i2 ... ina
TRANS     o1 o2 ... 1 g1 t1 ... gnt tnt
TRANP     o1 o2 ... 1 g1 t1 ... gnt tnt
PUSPL     a1 a2 1
LATCH     a1 a2 11 12
```

```
CAPCR     n1 n2
SOURC     n+
```

Appendix 3

Control Commands Used in Experimental Program PREMOS

1. TIME :

   General Form TIME Tstop Tstep Dtmin

   Tstop : the length of the analysis

   Tstep : output print step

   Dtmin : minimum internal timestep

2. PRESET :

   General Form PRESET (n1,v1) (n2,v2) ...

   n1,n2,... : node number

   v1,v2,... : preset node voltage

3. PLOT :

   General Form PLOT n1 n2 n3 ...

4. SEND :

   General Form SEND n1 n2 n3 ...

   The SEND command allows the user to generate the data file plfile.dat which contains the analysis results on node n1, n2 ... . The file plfile.dat is used as the input data for the graphing program graph.f.

5. global elements :

   General Form (type) (value)

   The global elements are

   (i) v+, the drain- or the load-end supply source

   (ii) v-, the source- or the driver-end supply source

(iii) vbg, the back gate supply voltage source

6. END :

General Form END

7. DC :

General Form DC

DC is the command requiring dc analysis.

8. CONTL :

General Form CONTL laten ltstp lpred

laten : flag of having latency scheme or not; 1 (yes), 0 (no)

ltstp : flag of having timestep control scheme or not;
1 (yes), 0 (no)

lpred : flag of having predictor scheme or not;
1 (yes), 0 (no)

9. OPT :

General Form OPT itnan itnor ittrs itpul itlch itao itoa

itnan : preset number of dc iterations for NAND2

itnor : preset number of dc iterations for NOR2

ittrs : preset number of dc iterations for TRANS

itpul : preset number of dc iterations for PUSPL

itlch : preset number of dc iterations for LATCH

itao : preset number of dc iterations for ANDOR

itoa : preset number of dc iterations for ORAND

Appendix 4

Analysis Data Structures for Subcircuit Models

The internal data structures representing the model of the subcircuit look like the following :

LINK

LOC(ISUB)	Number of Elements in This Field
	Model Type
	Pointer to Width List
	Pointer to Node List
	Feedback Node
	Pointer to Floating Capacitor List
	Others

The data structures for different types of subcircuits are listed below:

NAND2

LOC +0: 6
+1: 1
+2: INAN2
+3: IRAN2
+4: 0 or node number
+5: 0 or IFCAP

WNAN2

INAN2 +0: w/l (driver)
+1: w/l (load)

NAND2

IRAN2 +0: 1st input node

+1: 2nd input node

+2: internal node

+3: output node

NOR2

LOC +0: 6

+1: 2

+2: INOR2

+3: IROR2

+4: 0 or node number

+5: 0 or IFCAP

WNOR2

INOR2 +0: w/1 (driver)

+1: w/1 (load)

NOR2

IROR2 +0: 1st input node

+1: 2nd input node

+2: output node

ANDOI

LOC +0: 6

+1: 7

+2: IRAND

+3: IAND

+4: 0 or node number

+5: 0 or IFCAP

WAND

IRAND +0: w/1 (driver of AND)

+1: w/1 (driver of OR)

+2: w/1 (load)

NAND

IAND +0: na

+1: no

+2 -- +(na+1): a1 -- ana

+(na+2) -- +(na+no+1):

```
                                        o1 -- ono
                                +(na+no+2): output node
                                +(na+no+3) -- +(na+no+na+1):
                                        i1 -- i(na-1)

ORANI

      LOC +0: 6

          +1: 8                                 WOR
          +2: IROR                  IROR  +0: w/1 (driver of OR)
          +3: IOR                         +1: w/1 (driver of AND)
          +4: 0 or node number            +2: w/1 (load)
          +5: 0 or IFCAP

                                        NOR
                                IOR   +0: no
                                      +1: na
                                      +2 -- +(no+1): o1 -- ono
                                      +(no+2) -- +(no+na+1):
                                              a1 -- ana
                                      +(no+na+2): output node
                                      +(no+na+3) -- +(no+2na+2):
                                              i1 -- ina

TRANS

      LOC +0: 6

          +1: 4                                 WTFR
          +2: IRTFR                 IRTFR +0: w/1 (driver)
          +3: ITRF                        +1: w/1 (load)
          .
```

```
        +4: 0 or node number          +2: w/1 (transfer gate)
        +5: 0 or IFCAP

                                          NTFR
                                ITRF  +0: no
                                      +1: nt
                                      +2 -- +(jor+1): o1 -- ono
                                      +(jor+2): load node 1
                                      +(jor+3): gate node g1
                                      +(jor+4): source node t1
                                                :
                                                :
                                      +(jor+2nt+1): gate node gnt
                                      +(jor+2nt+2): output node tnt

PUSPL
      LOC +0: 6
          +1: 5                               WPUL
          +2: IWPUL                   IWPUL +0: w/1 (driver)
          +3: INPUL                         +1: w/1 (load)
          +4: 0 or node number
          +5: 0 or IFCAP                      NPUL
                                      NPUL  +0: driver gate node
                                            +1: load gate node
                                            +2: output node
```

LATCH

```
LOC +0: 6
    +1: 6                          WLH
    +2: IWLCH              IWLCH +0: w/l (driver)
    +3: ILCH                     +1: w/l (load)
    +4: 0 or node number
    +5: 0 or IFCAP                 NLH
                           ILCH  +0: 1st gate node
                                 +1: 2nd gate node
                                 +2: 1st output node
                                 +3: 2nd output node
```

SOURC

```
LOC +0: 4
    +1: 3                          NSOR
    +2: IVSC               IVSC  +0: node number
    +3: IRVSC                    +1: 0
                                 +2: 1

                                   WSOR
                           IRVSC +0:
                                 +1: TIME
                                 +2: VHIG
```

For floating capacitors, the data structures are:

CAPCR

NCAP

IFCAP +0: Node 1

+1: Node 2

CCAP

IFCAP +0: capacitor value

The order of node 1 and node 2 must coincide with the sequence of analysis so that the modified Gauss-Seidel method can be applied.

Appendix 5

Input Data File for The PLA Example

The input data file for simulating the PLA circuit in Fig. 7.1 by PREMOS is contained in this appendix.

```
PLA finite-state machine implementing the light controller
*subcircuit model card
model inv nor2 (5 1 10f 100f)
model nor3 andoi(5 5 1 10f 10f 10f 100f 0 3)
model nor4 andoi(5 5 1 10f 10f 10f 100f 0 4)
model notr1 trans(5 1 2 10f 100f 10f 50f 1 1)
model notr2 trans(5 1 2 10f 100f 10f 50f 2 1)
model notr4 trans(5 1 2 10f 100f 10f 50f 4 1)
model notr5 trans(5 1 2 10f 100f 10f 50f 5 1)
model clk1 source (4 1 10n 5n 10n 5n)
model clk2 source (5 0 5n 5n 5n 5n)
* AND plane
x1 11 17 19 1 nor3
x2 13 17 19 2 nor3
x3 12 14 17 19 3 nor4
x4 15 18 19 4 nor3
x5 16 18 19 5 nor3
x6 12 13 18 20 6 nor4
x7 11 18 20 7 nor3
x8 14 18 20 8 nor3
x9 15 17 20 9 nor3
x10 16 17 20 10 nor3
* OR plane
x11 5 6 7 8 9 21 56 28 notr5
x12 3 4 5 6 22 56 29 notr4
x13 3 5 7 8 10 23 56 30 notr5
x14 6 7 8 9 10 24 56 31 notr5
x15 4 5 25 56 32 notr2
x16 1 2 3 4 5 26 56 33 notr5
x17 9 10 27 56 34 notr2
* output registers
x18 28 35 55 49 notr1
x19 29 36 55 48 notr1
x20 30 30 37 inv
x21 31 31 38 inv
x22 32 32 39 inv
x23 33 33 40 inv
```

```
x24 34 34 41 inv
* input buffers
x25 57 42 55 45 notrl
x26 58 43 55 46 notrl
x27 59 44 55 47 notrl
* input registers
x28 45 45 50 inv
x29 46 46 51 inv
x30 47 47 52 inv
x31 48 48 53 inv
x32 49 49 54 inv
x33 50 50 11 inv
x34 45 45 12 inv
x35 51 51 13 inv
x36 46 46 14 inv
x37 52 52 15 inv
x38 47 47 16 inv
x40 53 53 17 inv
x41 48 48 18 inv
x42 54 54 19 inv
x43 49 49 20 inv
*input sources
va1 55 0 clk1 0 1 0 0 0 1 0 0 0 1 0 0 0 1
va2 56 0 clk1 0 0 0 1 0 0 0 1 0 0 0 1 0 0
va0 57 0 clk2 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1
vb0 58 0 clk2 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
vc0 59 0 clk2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
*analysis requests
opt 1 1 3 1 1 1 1
cont1 1 0 1
preset (35,5) (36,5)
time 120n 1n
plot 55 56 42 43 44 35 36
plot 37 38 39 40 41 9 10
plot 1 2 3 4 5 6 7 8
send 55 56 42 43 44 35 36
send 7 9 37 38 39 40 41
v+ 5
end
```

References

[1] W. W. Lattin, J. A. Bayliss, D. L. Budde, J. R. Rattner, and W. W. Richardson, "A Methodology for VLSI Chip Design," LAMBDA, Second Quarter 1981, pp. 34-44.

[2] A. R. Newton, "The Simulation of Large-Scale Integrated Circuits," ERL Memo No. ERL-M78/52, University of California, Berkeley, July 1978.

[3] R. E. Bryant, "An Algorithm for MOS Logic Simulation," LAMBDA, Fourth Quarter 1980, pp. 46-53.

[4] G. R. Case, "SALOGS-ACDC 6600 Program to Simulate Digital Logic Networks, Vol. 1 - User's Manual," Sandia Lab Report SAND 74-0441, 1975.

[5] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," ERL Memo., No. ERL-M520, University of California, Berkeley, May 1975.

[6] B. R. Chawla, H. K. Gummel and P. Kozak, "MOTIS - An MOS Timing Simulator," IEEE Trans. Circuits and Systems, Vol. CAS-22, No. 12, pp. 901-910, Dec. 1975.

[7] S. P. Fan, M. Y. Hsueh, A. R. Newton and D. O. Pederson, "MOTIS-C: A New Circuit Simulator for MOS LSI Circuits," Proc. IEEE Int. Symp. on Circuits and Systems, Phoenix, Arizona, pp. 700-703, April 1977.

[8] N. Tanabe, H. Nakamura and K. Kawkita, "MOSTAP: An MOS Circuit Simulator for LSI Circuits," Proc. IEEE Int. Symp. on Circuit and Systems, April, 1980, pp. 1035-1038.

[9] A. E. Ruehli, A. L. Sangiovanni-Vincentelli and N. B. Rabbat, "Time Analysis of Large-Scale Circuits Containing One-Way Macromodels," Proc. IEEE Int. Symp. on Circuits and Systems, April 1980, pp. 766-770.

[10] A. E. Ruehli, "Survey of Analysis, Simulation and Modeling for Large Scale Logic Circuits," 18th Design Automation Conference, Nashville, Tennessee, June 29 - July 1, 1981, pp. 124-129.

[11] Y. P. Wei, I. N. Hajj and T. N. Trick, "A Prediction-Relaxation-Based Simulator for MOS Circuits," Proc. IEEE Int. Conf. on Circuits and Computers, New York, Sept. 29 - Oct. 1, 1982.

[12] G. D. Hachtel and A. L. Sangiovanni-Vincentelli, "A Survey of Third-Generation Simulation Techniques," Proceedings of The IEEE, Vol. 69, No. 10, October 1981.

[13] H. Y. Hsieh and N. B. Rabbat, "Computer-Aided Design of Large Networks by Macromodular and Latent Techniques," Proc. IEEE Int. Symp. on Circuits and Systems, April 1977, pp. 688-692.

[14] P. Yang, "An Investigation of Ordering, Tearing and Latency Algorithms for The Time-Domain Simulation of Large Circuits," CSL Report R-891 University of Illinois at Urbana-Champaign,

August 1980.

[15] A. R. Newton, "Timing, Logic and Mixed-Mode Simulation for Large MOS Integrated Circuits," NATO Advanced Study Institute on Computer Design Aids for VLSI Circuits, Sogesta-Urbino, Italy, July 21 - August 1, 1980.

[16] J. M. Ortega and W. C. Rheinboldt, Iterative Solution of Non-linear Equations in Several Variables, New York, Academic Press, 1970.

[17] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli, "The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits," IEEE Trans. Computer-Aided Design, Vol. CAD-1, No. 3, pp. 131-145, July 1982.

[18] "HPMOTIS User's Manual," Hewlett Packard, October, 1981.

[19] Narsingh Deo, Graph Theory with Application to Engineering and Computer Science, Prentice Hall, 1974.

[20] Shimon Even, Graph Algorithms, Computer Science Press, Inc., 1979.

[21] A. V. Aho, J. E. Hopcroft and J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

[22] R. Tarjan, "Depth-First Search and Linear Graph Algorithms," SIAM J. Comput., Vol. 1, No. 2, pp. 146-160, June 1972.

[23] M. M. Denneau, "The Yorktown Simulation Engine," Proc. of 19th Design Automation Conference, Las Vegas, Nevada, June 14-16, 1982, pp. 400-409.

[24] D. E. Knuth, The Art of Computer Programming, Addison-Wesley, 1973.

[25] E. Horowitz and S. Sahni, Fundamentals of Data Structures, Computer Software Engineering Series.

[26] O. Wing and J. W. Huang, "A Computation Model of Parallel Solution of Linear Equations," IEEE Trans. Comput., Vol. C-29, pp. 632-638, July 1980.

[27] P. Roth, Computer Logic, Testing and Verification, Computer Science Press, Potomac, MD, 1980.

[28] C. V. Ramamoorthy, K. M. Chandy and M. J. Gonzalez, Jr., "Optimal Scheduling Strategies in A Multiprocessor System," IEEE Trans. Comput., Vol. C-21, pp. 137-146, Feb. 1972.

[29] S. Lam and R. Sethi, "Worst Case Analysis of Two Scheduling Algorithms," SIAM J. Comput., Vol. 6, No. 3, Sept. 1977.

[30] K. A. Sakallah, Mixed Simulation of Electronic Integrated Circuits, Ph.D. Thesis, Carnegie-Mellon University, 1981.

[31] R. S. Varga, Matrix Iterative Analysis, Prentice Hall, New Jersey, 1962.

[32] G. W. Stewart, Introduction to Matrix Computations, Academic Press, New York, 1973.

[33] Giovanni De Micheli and Alberto Sangiovanni-Vincentelli, "Numerical Properties of Algorithms for The Timing Analysis of MOS VLSI Circuits," European Conf. on Circuit Theory and Design, the Hague, Netherlands, Sept. 1981, pp. 387-392.

[34] L. O. Chua and P. M. Lin, Computer-Aided Analysis of Electronic Circuits, Prentice Hall, New Jersey, 1975.

[35] Dahlquist, Bjorck and Anderson, Numerical Methods, Prentice Hall, New Jersey, 1974.

[36] I. N. Hajj and S. Skelboe, "Time-Domain Analysis of Nonlinear Systems with Finite Number of Continuous Derivatives," IEEE Trans. Circuits and Systems, Vol. CAS-26, pp. 297-303, May 1979.

[37] M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, New York, Dover Publication, 1972.

[38] A. E. Ruehli, N. B. Rabbat, and H. Y. Hsieh, "Macromodular Latent Solution of Digital Networks Including Interconnections," Proc. 1978 IEEE Int. Symp. on Circuits and Systems, New York, May 17-19, 1978, pp. 515-521.

[39] Carver Mead and Lynn Conway, Introduction to VLSI Systems, Addison-Wesley, 1980.

[40] D. R. Alexander et al, "SPICE2 MOS Modeling Handbook," Report BDM/A-77-071-TR, BDM Corporation, Albuquerque, New Mexico, 1977.

[41] W. H. Change, "Analytical IC Metal-Line Capacitance Formulas," IEEE Trans. Microwave Theory and Techniques, pp. 608-611, September 1976.

[42] A. E. Ruehli and P. A. Brennan, "Accurate Metalligation Capacitances for Integrated Circuits and Packages," IEEE Journal of Solid State Circuits, pp. 289-290, August 1973

Vita

You-Pang Wei was born in Pong-Hu, Taiwan, China on July 6, 1953. He received his B.S. degree in electrical engineering from National Taiwan University in Taipei, Taiwan in June 1975. From 1975 to 1977 he was an ensign instructor in Chinese Navy, teaching several electronic courses. In August 1977 he entered the University of Illinois and received his M.S. degree one year later. During the academic year 1977-1979, he was a teaching assistant and research assistant in the Electrical Engineering Department and the Coordinated Science Research Laboratory, respectively. From August 1979 to January 1981, he worked for Advanced Micro Devices Inc. in Sunnyvale, California, as an integrated circuit design engineer. Since January 1981, he has held a teaching assistantship in the Electrical Engineering Department and a research assistantship with the Coordinated Science Research Laboratory. Presently he has accepted a position as senior CAD engineer in the CAD department of Intel Inc. in Santa Clara, California. His research interests are in the areas of computer-aided design, VLSI design, circuits and systems, semiconductor device modeling and solid state circuits.

END

DATE
FILMED

4-83

DTIC